第 1 章 Java 概述

1.1 Java 概述

1.1.1 什么是 Java

Java 是目前最为流行的编程语言之一，它是由 Sun 公司（目前被 Oracle 公司收购）于 1995 年推出，以其面向对象的特性而闻名，不仅继承了 C++语言的优点，同时摒弃了 C++难以理解的多继承和指针等概念，得到了广大编程人员的喜爱。Java 语言简单易用、安全可靠，在桌面应用程序、Web 应用程序和嵌入式系统应用程序开发等方面得到广泛应用。根据 2024 年 TIOBE 编程语言使用统计显示，Java 一直是最受欢迎的十款编程语言之一，如图 1-1 所示。

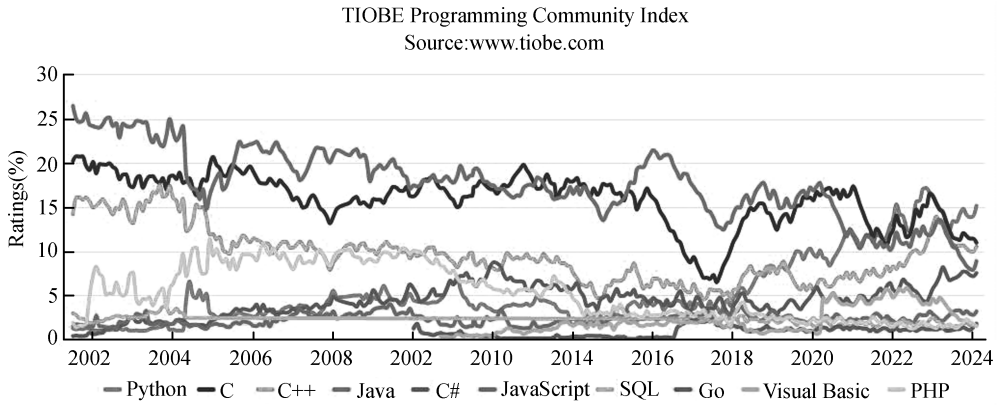


图 1-1 TIOBE 编程语言使用统计图

针对开发市场不同领域的需求，Java 分为 JavaSE、JavaEE 和 JavaME 三个技术平台。

- Java SE（Standard Edition）：Java 标准版平台，为开发普通桌面应用提供一个解决方案，也是三个平台中的核心部分。它为大多数个人 PC 上运行的 Java 应用程序的提供编

译和执行环境。

• Java EE（Enterprise Edition）：Java 企业版平台，为企业级应用程序开发提供解决方案。作为一个技术平台，它增加了一些关键的服务、协议和 API，如 Servlet、JSP、JavaBean、JDBC、EJB 和 Web Service 等技术。

• Java ME（Micro Edition）：Java 微型版，为小型数字电子设备和小型移动设备设计提供解决方案，它的目标是提供一种轻量级的 Java 实现，以便在这些设备上进行应用程序的开发。Java ME 支持 HTTP 等高级 Internet 协议，使得移动电话能够作为客户端和服务端直接访问互联网的信息，提高无线访问效率。

1.1.2 Java 特点

Java 语言是一个面向对象的高级程序设计语言，它之所以被广泛使用，Java 语言有很多突出的特点，其中主要有以下几个。

1. 安全性

Java 的安全性主要体现在存储分配模型、网络环境和运行时安全性。它采用了存储分配模型来防止恶意代码的攻击，并提供了字节确认器来确保代码的安全性。同时，Java 还采用了可执行代码安全性来保证程序的可靠性，并设置了访问控制机制来防止未经授权的访问。这些安全性机制使得 Java 成为一种安全可靠的编程语言。

2. 简单性

Java 语言的语法与 C 语言和 C++语言很相近，使得很多程序员学起来很容易。对 Java 来说，它舍弃了很多 C++中难以理解的特性，如操作符的重载和多继承等，而且 Java 语言不使用指针，加入了垃圾回收机制，解决了程序员需要管理内存的问题，使编程变得更加简单。

3. 面向对象

Java 提供了对类、对象、继承、封装、多态和接口等内容很好的支持。它只支持类之间的单继承，但支持接口之间的多继承可以使用接口来实现多继承。使用 Java 语言开发程序，需要采用面向对象的思想设计程序和编写代码。

4. 多线程

Java 的特点之一就是支持多线程，多线程是一种并发编程的方式，可以同时运行多个小任务，提高程序的并发性能和实时控制性能。Java 的 run( )方法可以同时运行多个线程，而不需要创建独立的线程对象。通过多线程，可以实现异步执行，同时可以同步执行多个小任务，提高程序的并发性能和实时控制性能。

5. 分布性

Java 是一种分布式的编程语言，它支持在网络上应用。Java 的分布性意味着它可以在多个不同的网络环境中运行，并且可以支持不同的数据源。这使得 Java 具有灵活性，适用于各种应用程序的需求。

6. 平台无关性

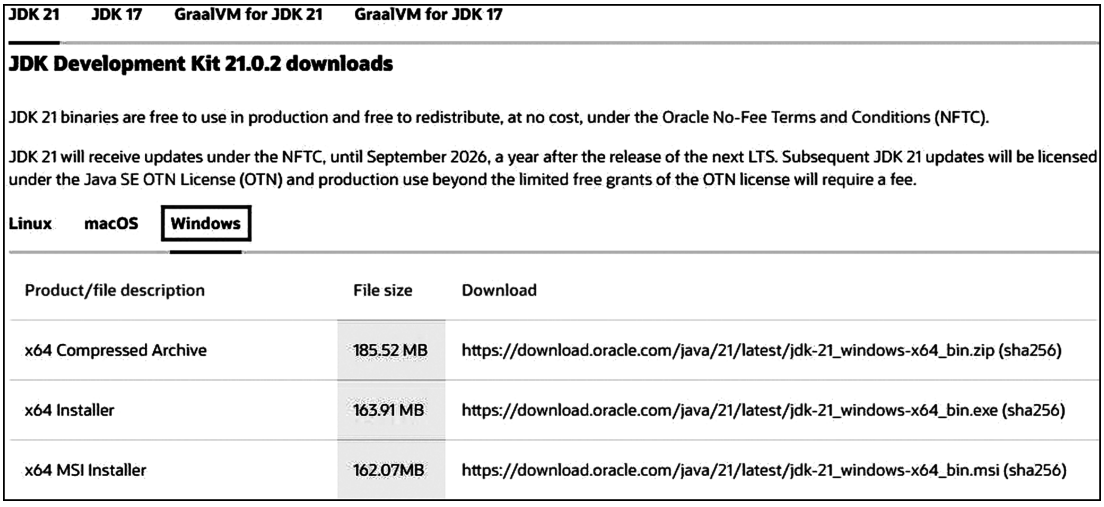
Java 平台无关性，也就是 Java 程序的可移植性，是指使用 Java 语言编写的程序可以在不同的计算机系统上运行，而不需要进行任何修改。这是因为 Java 编译成虚拟机后，只需要生成虚拟机上的目标代码，就可以在不同的平台上运行。而不会产生任何对平台无关性的影响。这种平台无关性可以大大降低开发、维护和管理的开销，提高开发效率。

1.2　JDK 的下载与安装

想要进行 Java 代码的编译、执行及其他操作，需要在本地安装 Java 开发环境。Oracle 公司提供了开发者工具包，简称 JDK（Java Development Kit），包括包括 Java 编译器、Java 虚拟机、Java API 类库等。下面以 64 位 Windows10 系统为例，介绍 JDK-21 的下载与安装过程，步骤如下。

1.2.1　下载 JDK-21

访问 Oracle 公司官网（https://www.oracle.com/java/technologies/downloads/），下载 JDK 安装文件“jdk-21_windows-x64_bin.exe”。在这里，可以根据需要，选择 Linux、macOS 以及 Windows 操作系统所对应的 JDK 安装程序。如图 1-2 所示。


JDK 21　JDK 17　GraalVM for JDK 21　GraalVM for JDK 17

JDK Development Kit 21.0.2 downloads

JDK 21 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions (NFTC).

JDK 21 will receive updates under the NFTC, until September 2026, a year after the release of the next LTS. Subsequent JDK 21 updates will be licensed under the Java SE OTN License (OTN) and production use beyond the limited free grants of the OTN license will require a fee.

Linux　macOS　Windows

Product/file description	File size	Download
x64 Compressed Archive	185.52 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.zip (sha256)
x64 Installer	163.91 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe (sha256)
x64 MSI Installer	162.07MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.msi (sha256)

图 1-2　JDK 下载页面

1.2.2　安装 JDK-21

双击下载好的 JDK 安装文件，进入到 JDK-21 的安装界面，如图 1-3 所示。

图 1-3　JDK-21 安装界面

在图 1-3 中，单击【下一步】按钮进入到 JDK 安装目录选择界面，如图 1-4 所示。

图 1-4　JDK-21 安装目标文件夹

在图 1-4 中，根据用户需要，单击【更改】按钮，更改 JDK 安装文件夹。完成更改后，单击单击【下一步】按钮，安装程序自动安装，直至完成，如图 1-5 所示。

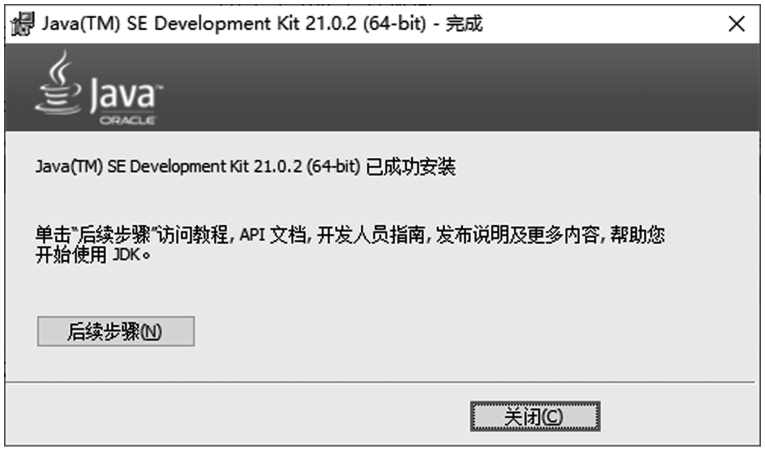


图 1-5　JDK-21 完成安装

1.3　环境变量配置

在进行 Java 源程序编译和运行等操作时，需使用 JDK 中的 javac. exe 和 java. exe 等工具，若不配置环境变量，只能将 Java 源程序放置于编译与运行工具同目录下，操作不便；配置环境变量后，可以在任何目录下运行 Java 源程序。下面将详细介绍环境变量的配置过程。

右键单击【开始】，选择【系统】，弹出系统设置窗口，如图 1-6 所示；

图 1-6　系统设置窗口

在系统设置窗口，单击【高级系统设置】，弹出系统属性窗口，选择【高级】选项卡，如图 1-7 所示；单击【环境变量】，弹出环境变量设置对话框，如图 1-8 所示。

图 1-7　系统属性

图 1-8　环境变量

下面开始配置 Java 运行环境中需要配置的 JAVA_HOME、Path、CLASSPATH 这三个环境变量：

首先，单击图 1-8 中系统变量中的【新建】按钮，弹出如图 1-9 所示对话框，设置变量名为 JAVA_HOME，变量值为本地 Java 安装目录，然后单击【确定】按钮即可。

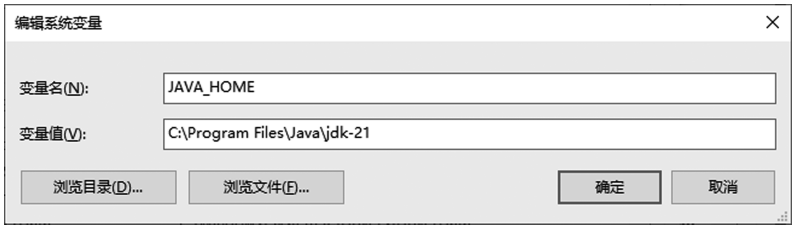


图 1-9　JAVA_HOME 环境变量设置

其次，再次单击图 1-8 中系统变量中的【新建】按钮，弹出如图 1-10 所示对话框，设置变量名为 CLASSPATH，设置变量值为 .;%JAVA_HOME%\lib;%JAVA_HOME%\lib\tools.jar，然后单击【确定】按钮即可。

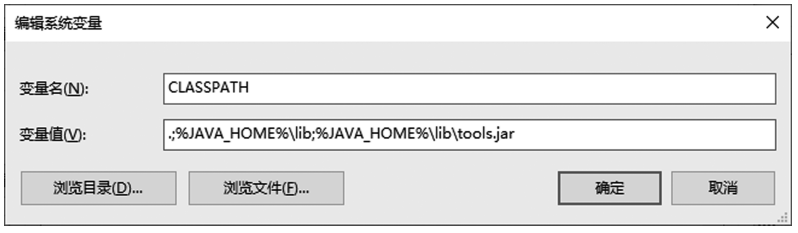


图 1-10　CLASSPATH 环境变量设置

最后，在图 1-11 系统变量中找到 Path 变量，单击【编辑】按钮，弹出如图 1-11 所示对话框，单击【新建】按钮，输入%JAVA_HOME%\bin，单击【确定】按钮，完成设置，如图 1-12 所示。

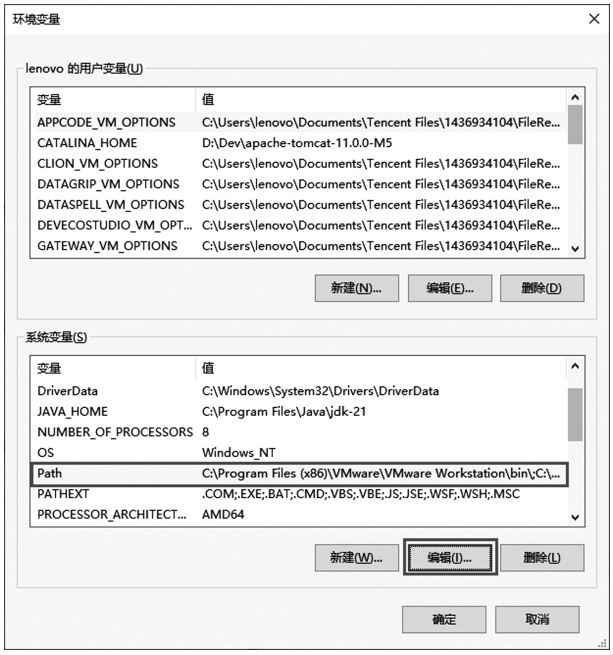


图 1-11　环境变量

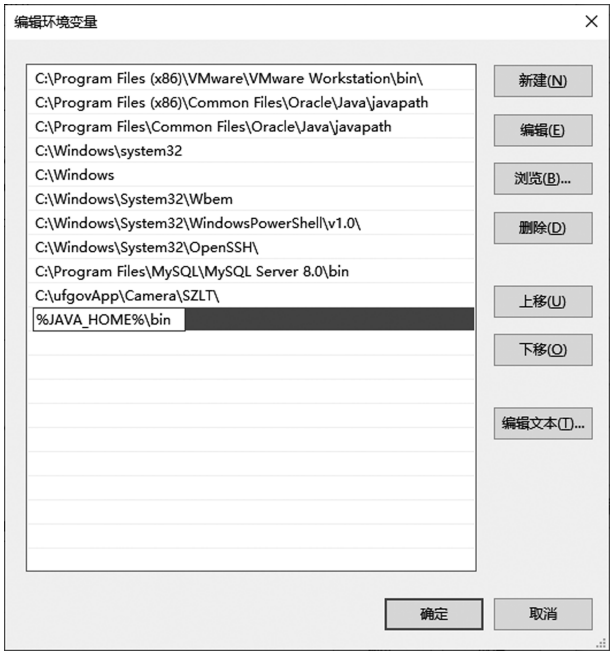


图 1-12　Path 环境变量设置

至此，JDK 环境变量设置完毕，Java 源代码可以在本地环境任何目录下进行编译和解释执行了。

1.4 第一个 Java 程序

配置好 JDK 环境变量后，就可以进行 Java 代码的编写和运行了。下面通过一个案例来详细讲述 Java 代码的编写与运行过程。

1.4.1 Java 源代码编写

首先，在本地电脑 F 盘根目录（根据自己本地计算机自行选择合适目录）下中新建一个文件夹 test（方便管理 Java 源文件），在此文件夹中新建一个文本文档，重命名为 Demo0101.java，注意要将文本文件的扩展名 .txt 修改成 Java 文件扩展名 .java。文件 Demo0101.java 代码如下所示。

```
public class Demo0101 {
    public static void main(String[ ] args) {
        System. out. println("This is the first Java program!");
    }
}
```

文件 Demo0101.java 代码主要功能就是向控制台（屏幕）输出打印“This is the first Java program!”这个字符串，下面详细解释这段 Java 代码。

- 第一行代码：public class Demo0101，在这里定义了一个公共的类 Demo0101。public 声明的是这个类是公共类，class 是定义类的关键字，Demo0101 是这个类的名字。在这里需要注意的是，每个 Java 源文件中，最多有一个公共类，一旦某个类被声明成公共类，这个类所在的 Java 文件的名称要与类名保持一致；Demo0101 作为类的名称，建议首字母大写，方便与变量名进行区分；在类名后有一对大括号{}，大括号内部便是类的功能实现的地方，类的主体部分，可以简单成为类体。
- 第二行代码：public static void main（String[ ] args），定义了静态的、公共的主方法 main。public 用于声明此方法为公共方法，static 声明方法为静态方法，void 声明了方法的返回值类型，main 是方法的名称，String[ ] args 定义了字符串数组作为 main( )方法的形式参数。main( )作为 Java 程序执行的入口，程序将从 main( )后面的大括号{}内部的代码开始执行。
- 第三行代码：System. out. println（“This is the first Java program!”）;，它的作用是实现了向控制台打印输出“This is the first Java program!”。

在进行代码编写时，特别要注意的是，程序源代码中涉及到的括号、分号等字符要采

用英文半角格式，要严格区分字母的大小写，它们表达的含义不一样。使用记事本编写的第一个 Java 程序如图 1-13 所示。

图 1-13　记事本编写 Java 代码

1.4.2 Java 源代码编译

第一，通过命令进入指定目录。F：+enter 进入 F 盘根目录，cd test 进入 F 盘下面的 test 目录，如图 1-14 所示。

图 1-14　进入源文件目录 test

第二，在 test 后面的命令提示符输入“javac Demo0101.java”，然后回车执行命令，完成 Java 源文件的编译，如图 1-15 所示。

图 1-15　编译 Demo0101.java 文件

javac 是编译工具，通过编译后，将源程序编译成字节码文件（.class 文件），此时在 test 目录中就会有一个名为 Demo0101.class 的字节码文件，如图 1-16 所示。

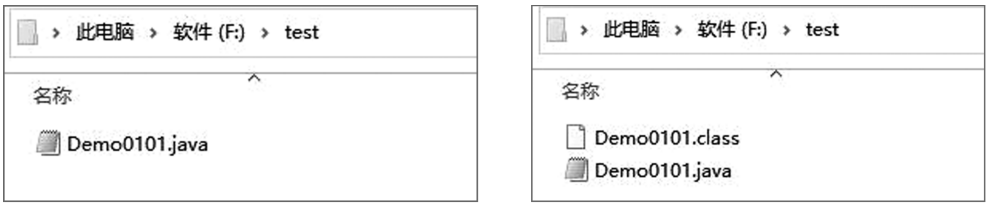


图 1-16　编译前后 test 目录

1.4.3 运行 Java 程序

编译完成后，便可以使用解释执行工具 java，对字节码文件 Demo0101.class 进行解释执行了。在命令提示符后，输入“java Demo0101”命令，输入回车，执行命令，最终程序运行结果“This is the first Java Program !”输出打印在屏幕上，如图 1-17 所示。

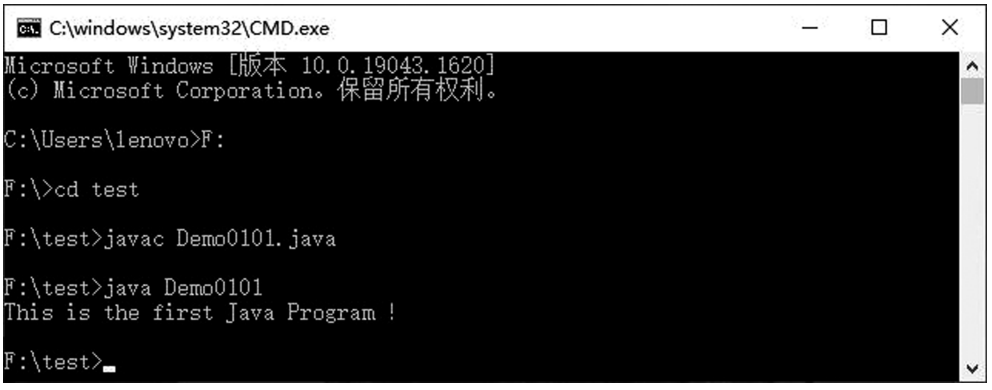


图 1-17　文件 Demo0101 运行结果

1.5 Eclipse IDE

1.5.1 Eclipse 的下载与安装

登录 Eclipse 官方网站下载页面（https://www.eclipse.org/downloads/packages/），根据需要选择合适的版本。若只进行桌面级开发，选择 Eclipse IDE for Java Developers 版本；若要进行企业级开发，选择 Eclipse IDE for Enterprise Java and Web Developers 版本。本书下载的是基于 Windows 系统的 Eclipse IDE Packages，如图 1-18 所示。

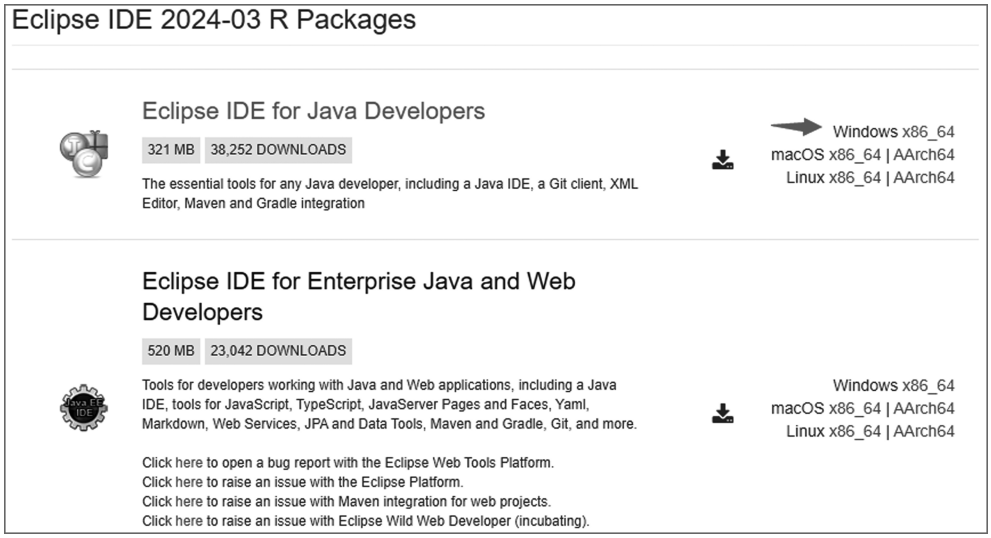


图 1-18　Eclipse IDE 下载页面

下载完成后，根据个人需求，将 Eclipse IDE Packages 压缩包放置本地电脑文件夹中，并解压压缩包，打开 eclipse 文件夹，直接双击 eclipse. exe 或者将其发送到桌面再启动，如图 1-19 所示。

图 1-19　Eclipse 压缩包

Eclipse 启动之后，会弹出工作空间（Java 工程存放的目录）选择对话框。如需更改，单击【Browse】按钮，选择需要的目录位置。每次启动 Eclipse，都会有选择工作空间对话框，若确定好工作空间后暂时不更改，可以勾选 Use this as the default and do not ask again 复选框，这样 Eclipse 就默认工作空间，再次启动不再出现选择工作空间对话框。如图 1-20、1-21 所示。

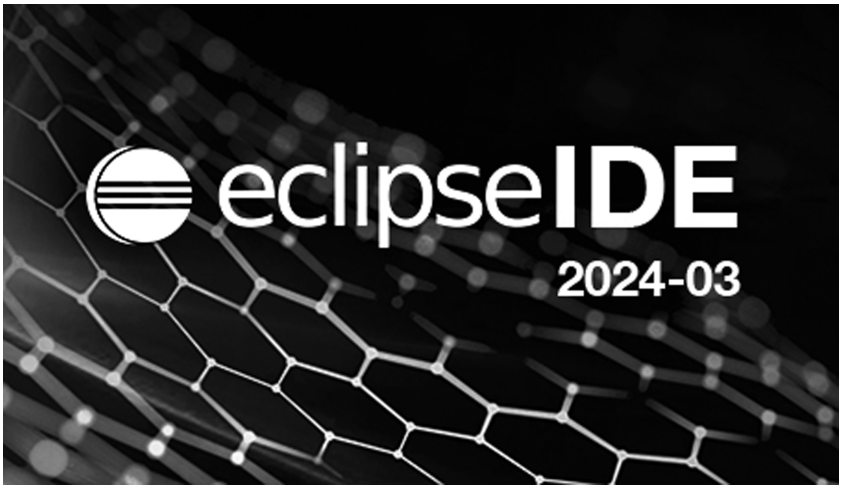


图 1-20　Eclipse 启动界面

图 1-21　选择工作空间对话框

1.5.2 Eclipse 使用

Eclipse 启动后，主界面如图 1-22 所示。与其他软件类似，最上面是菜单栏，菜单栏下面是工具栏，再下面便是工作区。

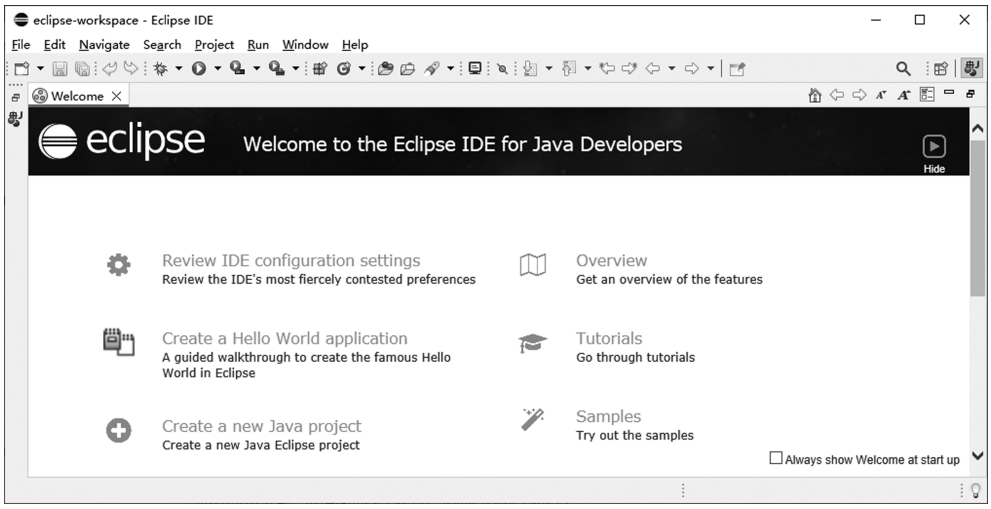


图 1-22　Eclipse 主界面

下面通过编写 Java 程序，来介绍 Eclipse 的基本应用。

首先，单击【File】菜单，选择【new】命令，在下一级菜单选择【Java Project】，弹出新建工程的对话框，输入工程的名字，单击【Finish】按钮，完成工程的创建，如图 1-23 所示。

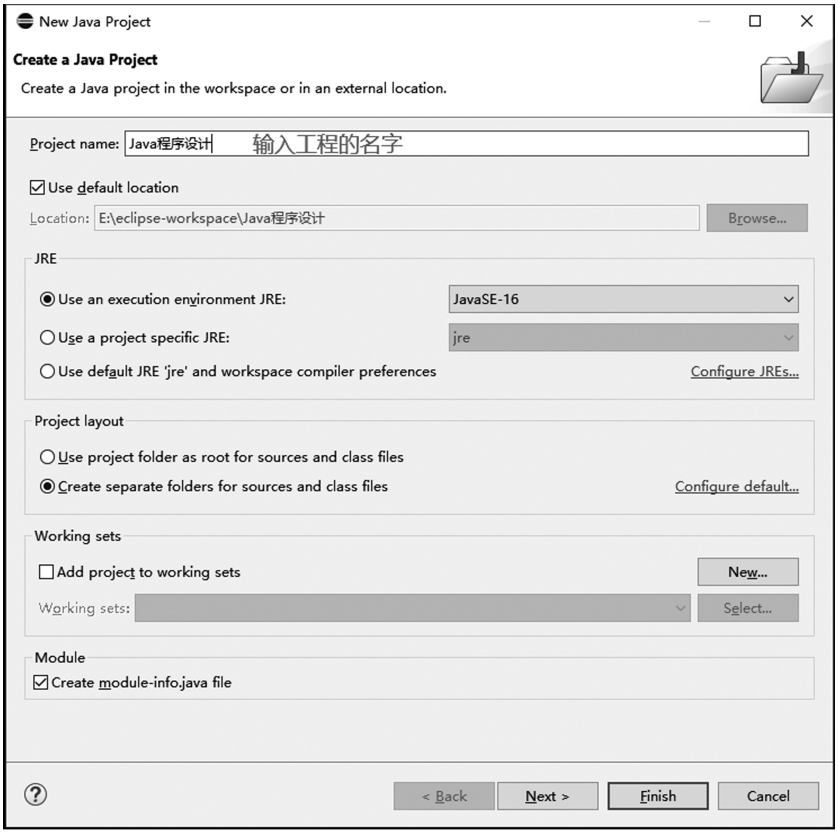


图 1-23　新建 Java 工程

其次，在工程中，右键单击【new】命令，在下一级菜单选择【class】，新建一个类，输入类的名字，可勾选 public static void main（String[ ]args）复选框，自动创建主方法，单击单击【Finish】按钮，完成类的创建，如图 1-24 所示。

图 1-24　新建一个类

在代码编辑区域，编写 Java 程序代码，代码如图 1-25 所示。

```
Demo0102.java
1 package ch01;
2 public class Demo0102 {
3     public static void main(String[] args) {
4         // TODO Auto-generated method stub
5         System.out.println("Hello world!");
6     }
7 }
```

图 1-25　Java 源代码

在文件 Demo0102 代码中，编写了 Java 的主方法 main，它是程序执行的入口；在 main 方法中，编写了一条输出语句。文件 Demo0102 运行结果如图 1-26 所示。

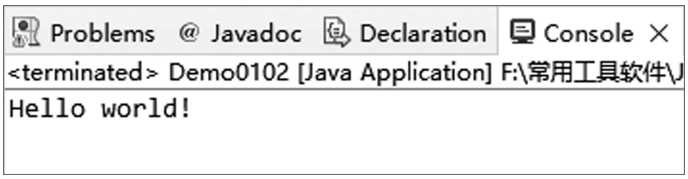


图 1-26　文件 Demo0101 运行结果

1.6　IntelliJ IDEA

IntelliJ IDEA 简称 IDEA，是 Java 编程的集成开发环境，深受广大开发者的喜爱。IntelliJ IDEA 分为 Ultimate Edition 旗舰版和 Community Edition 社区版本，旗舰版需要付费购买，社区版本免费使用，但是功能上对比旗舰版有所缩减。

1.6.1　下载与安装

登录 IntelliJ IDEA 的官网 https：//www. jetbrains. com/idea/download/other. html，根据本地操作系统，选择合适的安装版本进行。本书以社区版为例，对 IntelliJ IDEA 的下载与安装进行详细将讲解。IntelliJ IDEA 各版本的下载页面如图 1-27 所示。

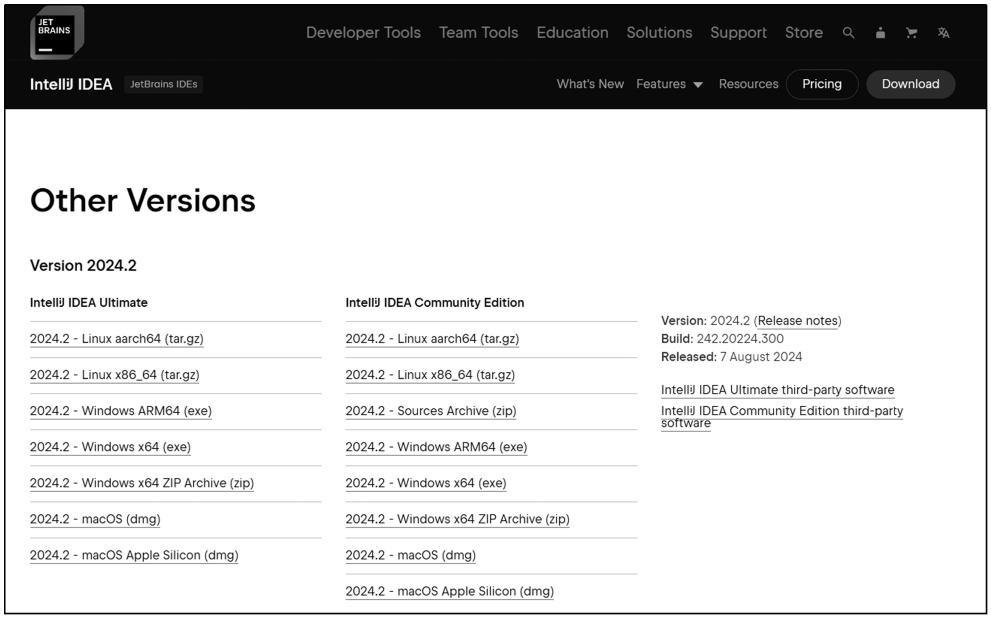


图 1-27　IntelliJ IDEA 下载页面

下载完成后，便可以进行安装。本书下载的是社区版 ideaIC-2024.2.exe 安装程序，进入到 IntelliJ IDEA 社区版的安装向导界面，如图 1-28 所示。

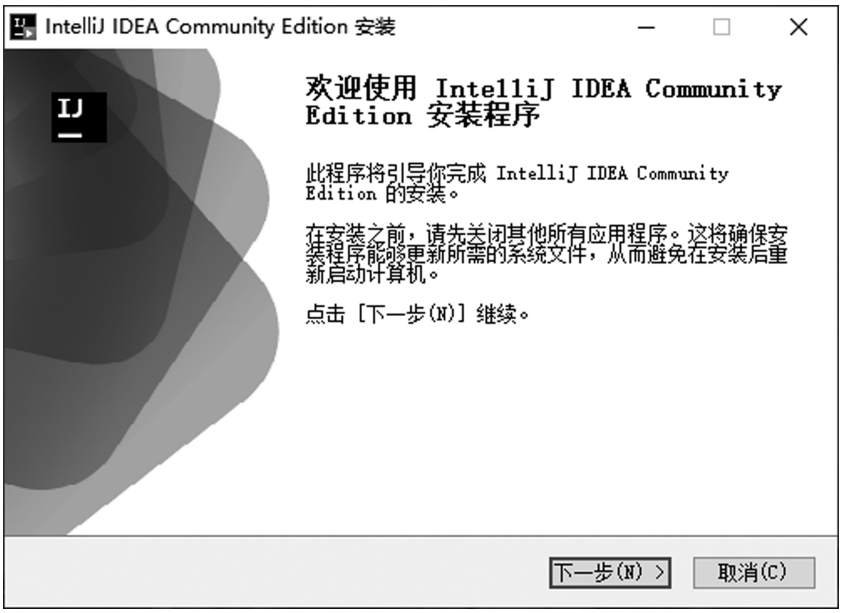


图 1-28　IntelliJ IDEA 社区版安装

在图 1-28 中，单击【下一步】按钮，进入到安装位置选择界面，可以默认安装，也可以单击【浏览】按钮，选择合适的安装目录，如图 1-29 所示。

图 1-29　安装目录选择

在图 1-29 中，单击【下一步】按钮，进入到安装选项选择界面，可以根据自己需求勾选复选框，建议勾选最上面两个复选框，如图 1-30 所示。

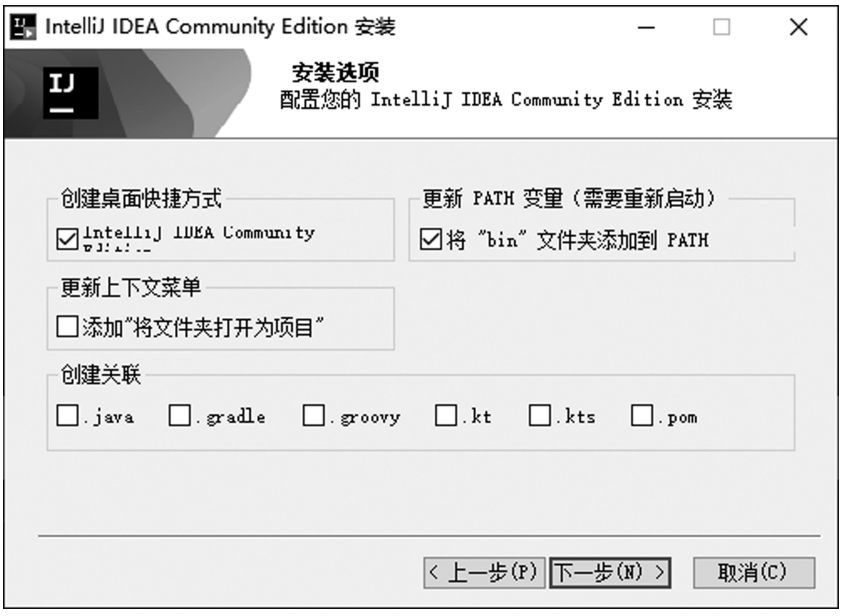


图 1-30　安装选项

在图 1-30 中，单击【下一步】按钮，进入到选择开始菜单文件夹界面，默认安装，也可以自定义名称，如图 1-31 所示。

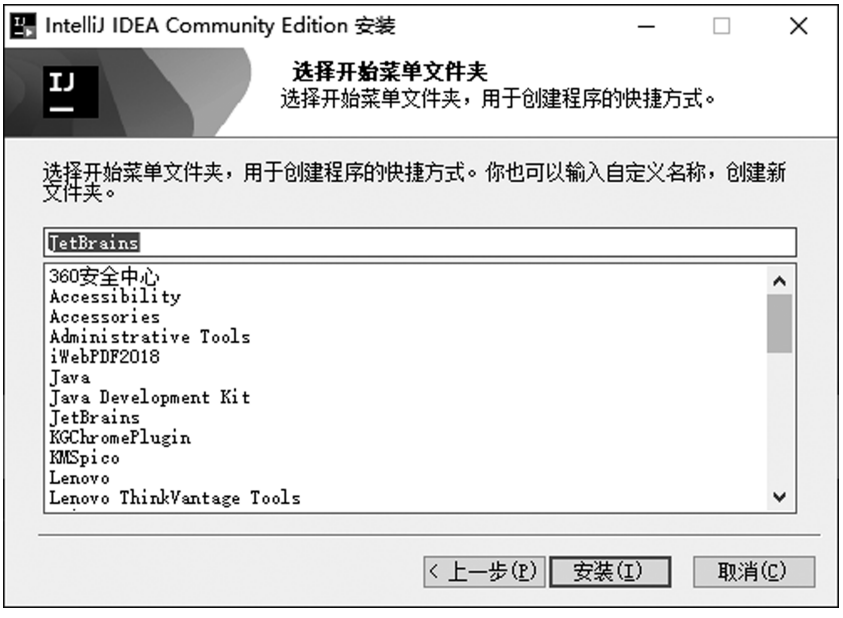


图 1-31　选择开始菜单文件夹

在图 1-31 中，单击【安装】按钮，进入到 IntelliJ IDEA 社区版过程中去，可以查看安装进度与安装详情，如图 1-32 所示。

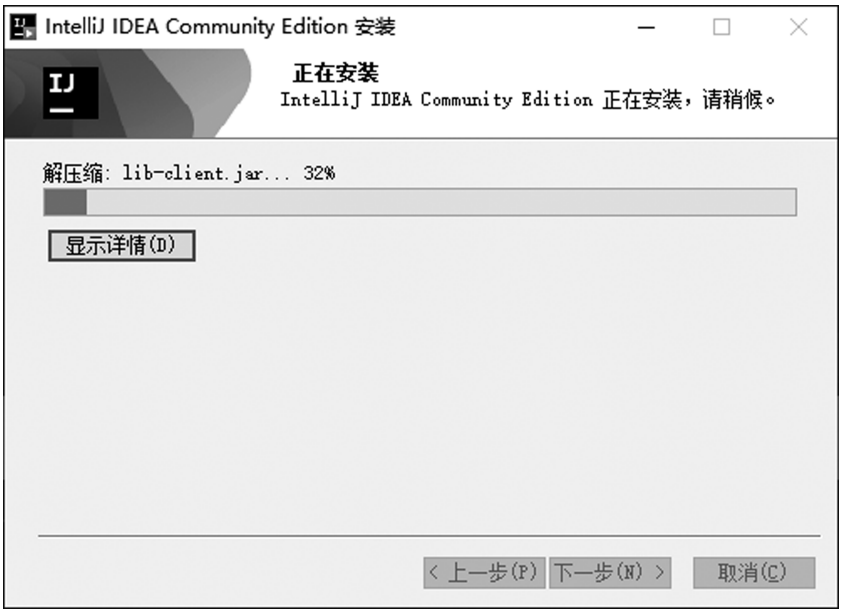


图 1-32　正在安装界面

在图 1-32 中，等待安装完成，进入安装结束界面，可以选择立即重启，也可选择稍后重启，如图 1-33 所示。

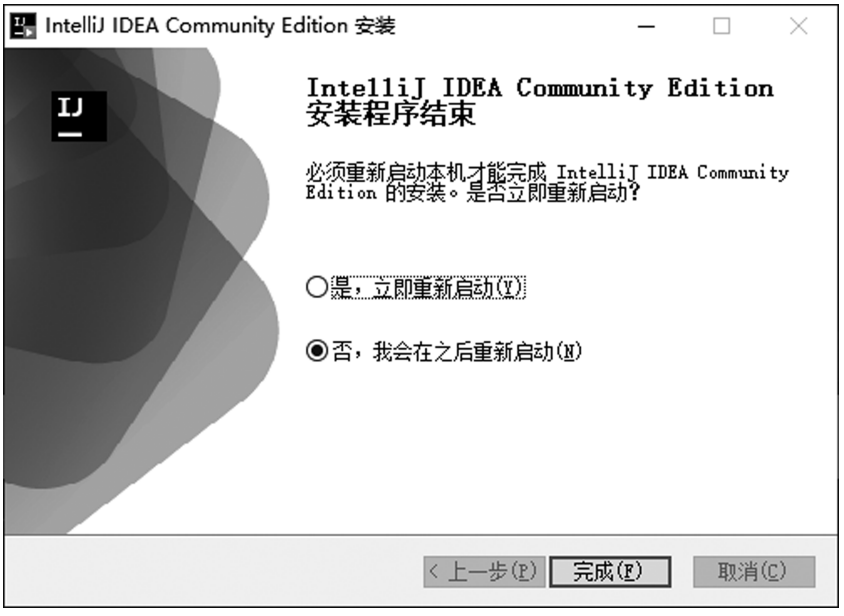


图 1-33　安装结束界面

1.6.2 IntelliJ IDEA 使用

安装完成后，就可以使用 IntelliJ IDEA 进行 Java 代码开发了，下面介绍使用 IntelliJ IDEA 进行 Java 代码开发的基本过程。

在本地电脑桌面上找到 IntelliJ IDEA Community Edition 2024.2 图标，双击图标启动 IntelliJ IDEA，启动界面如图 1-34 所示。

图 1-34　IntelliJ IDEA 启动界面

启动后，进入到用户协议界面，勾选已阅读并接受此《用户协议》复选框，如图 1-35所示。

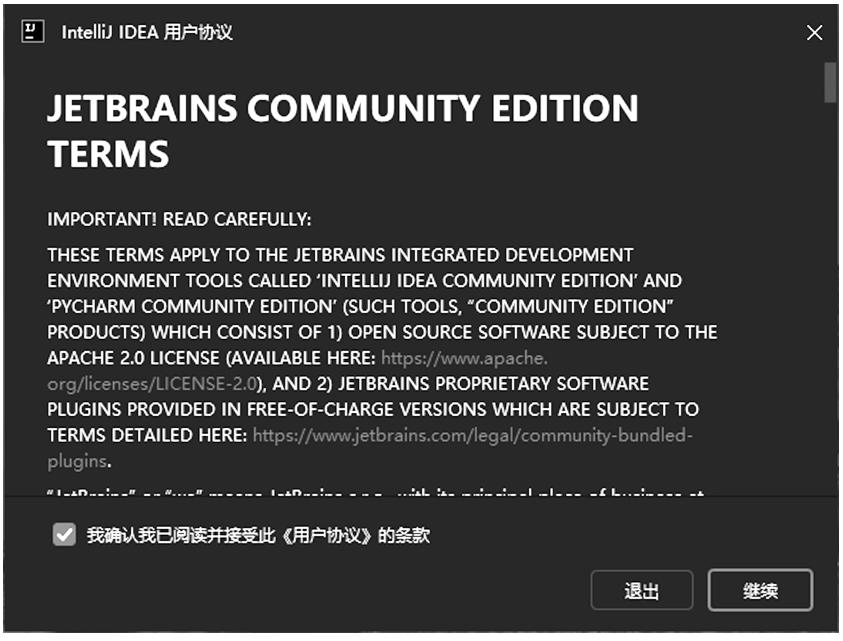


图 1-35　用户协议界面

在图 1-35 中，单击【继续】按钮，进入到 IntelliJ IDEA 欢迎界面，在这里单击【New Project】按钮新建工程；也可以单击【Open】按钮，打开已有工程，如图 1-36 所示。

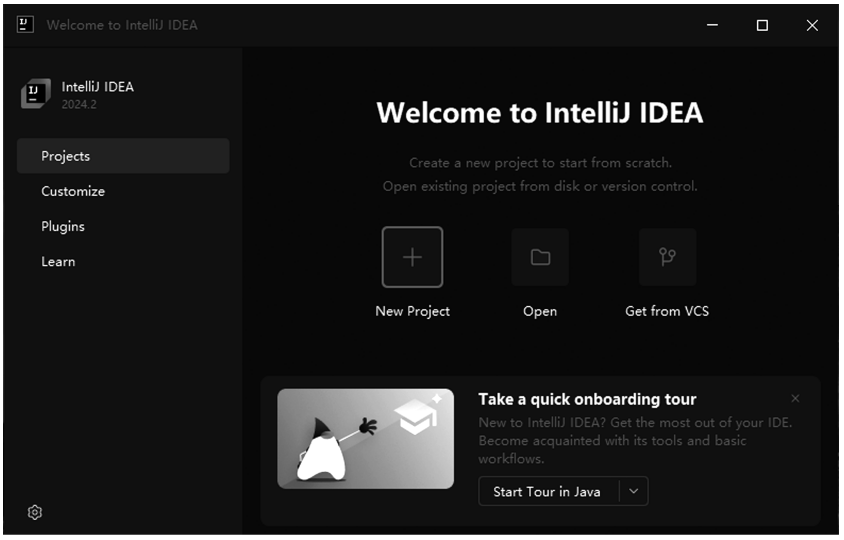


图 1-36　IntelliJ IDEA 欢迎界面

下面新建一个 Java 工程，单击图 1-36 中的【New Project】按钮，进入到新建工程页面，选择页面左侧【Java】工程。在页面右侧【Name】标签右侧文本框为新建的工程填写工程名字；在【Build system】处，选择【IntelliJ】选项；若系统装有多个版本的 JDK，可以根据需要进行选择，如图 1-37 所示。

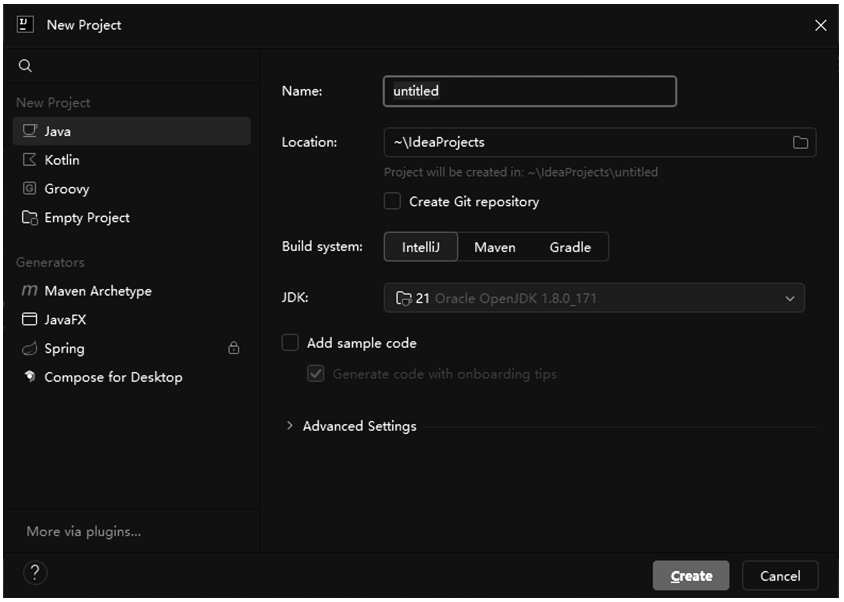


图 1-37　新建工程

新建工程后，在左侧工程管理器下面的 src 包处右键单击，弹出第一级菜单，选择【New】命令，弹出第二级菜单，如图 1-38 所示。

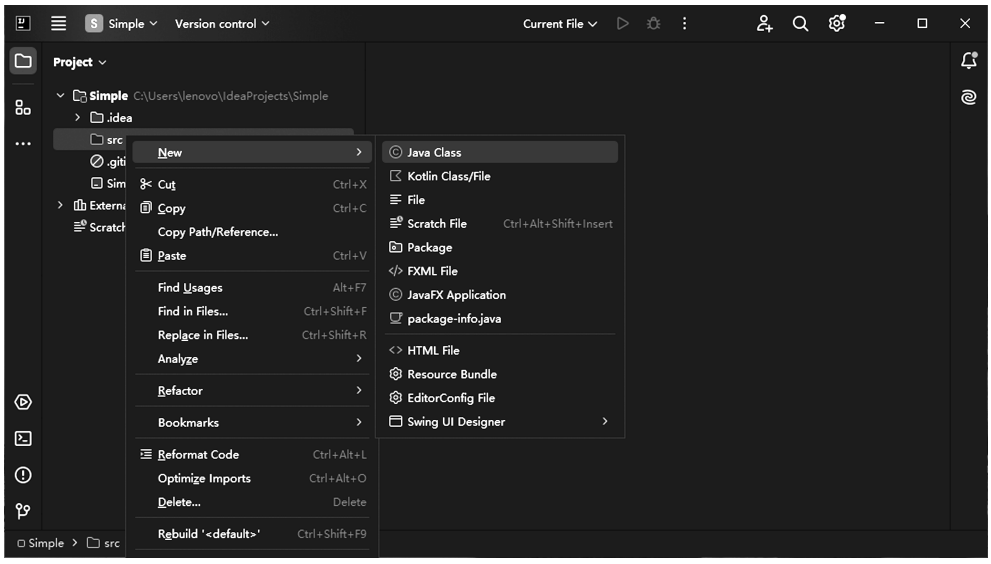


图 1-38　工程新建文件

在图 1-38 界面弹出第二级菜单上，单击【Java Class】命令，输入要新建的 Java 类的名称，并可以选择要新建 Java 类的类型（Class 类、Interface 接口、Enum 枚举等），如图 1-39所示。

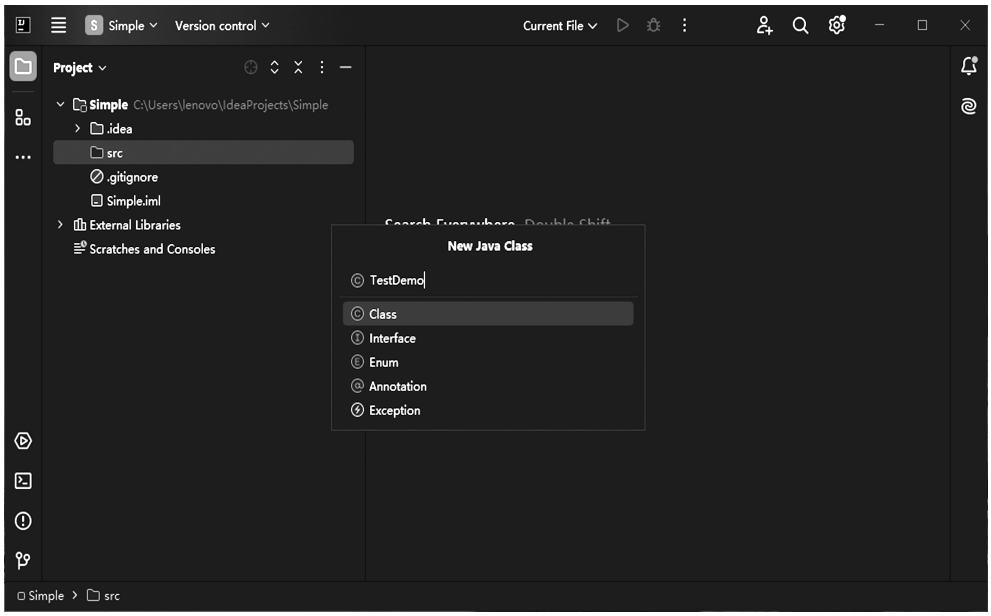


图 1-39　新建 Java 类

新建完成 Java 类后，便可以在右侧代码编辑区域进行代码的编写，在这里编写能够输出“Hello world!”的程序，如图 1-40 所示。

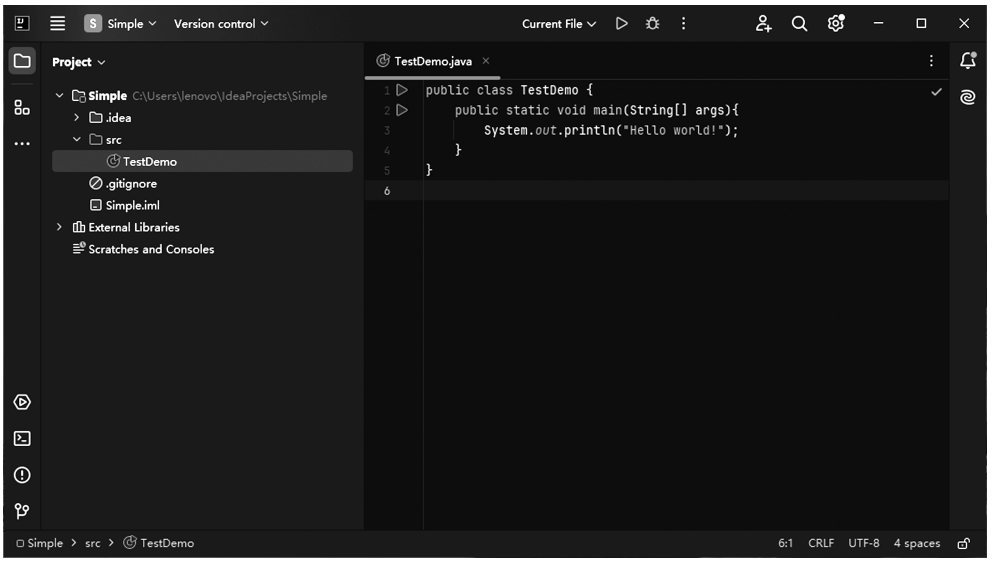


图 1-40　Java 代码编辑

编辑好代码后，单击  顶部的绿色三角，编译运行写好的代码，结果显示在控制台上，如图 1-41 所示。

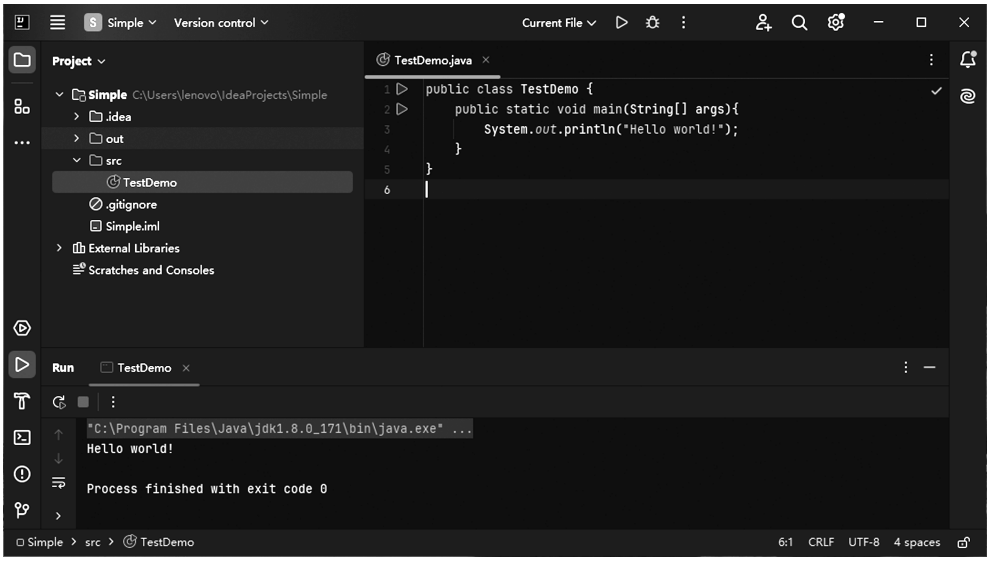


图 1-41　编译运行 Java 程序

以上是 IntelliJ IDEA 的下载与安装，以及使用 IntelliJ IDEA 进行 Java 程序的编写的详细步骤，大家可以根据以上步骤进行代码编写的练习。

练习题

一、填空题

1. Java 源程序文件的后缀是_____，Java 字节码文件的后缀名称是_____。
2. Java 程序实现可移值性，依靠的是_____。
3. Java 语言的三个分支是：_____、_____、_____ 。
4. Java 程序由_____组成，如果 Java 使用_____声明类，则文件名称必须与类名称一致。
5. Java 执行是从_____ 方法开始执行的，此方法的完整定义是_____。
6. 从命名标准上来讲，Java 类名的每个单词的首字母通常要求_____ 。

二、选择题

1. 编译 Java Application 源文件将产生相应的字节码文件，扩展名是（　　）。

   A. . Java　　B. . class　　C. . html　　D. . exe

2. Java 语言具有许多优点和特点，下列选项中，（　　）反映了 Java 中“一次编译，随处运行”的特点。

   A. 简单性　　B. 平台无关性　　C. 面向对象性　　D. 安全性

3. Java 语言具有许多优点和特点，下列选项中，（　　）反映了 Java 中并行机制的特点。

   A. 多线程　　B. 平台无关性

   C. 面向对象性　　D. 安全性

4. 在 Java 语言中，下列（　　）语句关于内存回收的说明是正确的？

   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. 内存回收程序可以在指定的时间释放内存对象

5. 一个 Java 源文件中最多只能有（　　）public 类。

   A. 1 个　　B. 2 个　　C. 3 个　　D. 任意多个

6. 下面（　　）不是 Java 语言所具有的特点。

   A. 没有全局变量，在类的定义外部没有任何的变量定义

   B. 不直接支持指针操作，而使用更安全的引用类型

   C. 支持子类对父类的多继承

   D. 具有自动垃圾收集功能

7. 下面（　　）不是 Java 开发工具包 JDK 的组成部分。

A. Java 的编译器　　B. Java 的解释器

C. Java 的 API 继承类库　　D. Eclipse 开发工具

8. JDK 提供的编译器是（　　）。

A. Java. exe　　B. Javac. exe

C. Javap. exe　　D. Javaw. exe

9. 作为 Java 应用程序入口的 main 方法，其声明格式可以是（　　）。

A. public void main（String[ ]args）

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

C. public static void Main（String * args）

D. public int main（String[ ]args）

10. 下列说法正确的是（　　）。

A. Java 程序的 main 方法必须写在类里面

B. Java 程序中可以有多个 main 方法

C. Java 程序中类名必须与文件名一样

D. Java 程序的 main 方法中如果只有一条语句，可以不用 {}（大括号）括起来

11. 下列哪些语句关于 Java 内存回收的说明是正确的？（　　）

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. 内存回收程序可以在指定的时间释放内存对象

三、判断题

1. Java 语言属于编译型的开发语言。（　　）
2. Java Application 程序不是由 main( )方法开始执行的。（　　）
3. Java 语言属于高级语言。（　　）
4. Java 语言与 C 语言一样都是面向过程的语言。（　　）

四、简答题

1. 简述 Java 程序平台无关性的基本原理。
2. 简述 Java 语言的主要特点。
3. 编写程序，实现“This is a Java program!”字符串的输出。
4. 编写程序，在屏幕上打印出以下图形。

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * *   Java 程序设计   * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

第 2 章 Java 基本语法

2.1 标识符

标识符可以用于类、属性、方法等其名称的定义。Java 中标识符由字母、数字、下划线（_）、美元符号（$）组成，且不能以数字开头，也不能是 Java 中的关键字。

下面为合法标识符。

```
$name
_user
stuName
Case
```

下面为不合法标识符。

```
break
123user
- name
int
```

标识符要严格按照以上规则命名，但同时为了增加程序的可读性，尽量遵守以下规则（不是强制规则）：

- 变量名、方法名第一个单词首字母要小写，第二个单词开始，首字母要大写，例如：userName、getName 等；
- 类名、接口名每个单词首字母都要大写，第二个单词开始，首字母要大写，例如 StuInfo、DataUtil；

变量名、方法名、类名以及接口名的命名规则也成为驼峰命名法。

- 常量名要全部大写，每个单词要用下划线分隔，例如 USER_NAME、MATH_PI 等；
- 包名全部小写，例如 hy. cn；
- 对标识符进行命名，尽量做到见名之意。

2.2 关键字

与其他语言一样，在 Java 中有一组事先定义好有特殊意义的单词，把它们称作关键字，有的书中也称作保留字。

在进行 Java 编程时，要注意所有关键字都是英文小写；关键字不能在用作变量、类、方法等其名称的定义。true、false 与 null 虽然不是关键字，但在 Java 中已近赋予特殊的意义，也不能用作标识符的命名。Java 中的关键字如表 2-1 所示。

表 2-1 Java 关键字

abstract	assert	boolean	break	byte	case	catch
char	class	continue	const	default	do	double
else	extends	enum	final	finally	float	for
goto	if	implements	import	instanceof	int	interface
long	native	new	package	private	protected	public
return	short	static	synchronized	super	strictfp	this
throw	throws	transient	try	void	volatile	while

2.3 注释

在编写 Java 程序时，要及时对代码进行必要的注释，增加程序的可读性以及可维护性，这些注释不会被编译器进行编译。所以，注释的使用，不仅可以对代码进行解释说明，还可以在进行调试程序时，将暂时不需要执行的代码进行注释，使这段代码不参与程序执行，帮助开发人员确定 bug 所在的地方。

在 Java 中，注释一共分为单行注释、多行注释以及文档注释三种，下面分别对这三种注释形式进行讲解。

2.3.1 单行注释

单行注释主要用于代码中一行语句的的解释说明，用符号“//”表示，符号“//”后面就可以写要解释说明的内容，如下面代码段所示。

```
int age=18;                    //定义整型变量 age,并赋值 18
double score=96.5;             //定义双精度型变量 score,并赋值 96.5
String name="李想";            //定义字符串变量 name,并赋值"李想"
```

2.3.2 多行注释

多行注释简单的说就是对程序的多行语句进行注释，以符号“/*”开始，并以“*/”结束，形式如下面代码所示。

```
/* 声明一个整型数组 array
     为这个整型数组开辟 5 个空间 */
int[ ] array = null;
array = new int[5];
```

2.3.3 文档注释

文档注释通常是对程序中的某个类、方法、接口等结构进行全面的解释说明，文档注释以“/**”开始，以“*/”结束，形式如下面代码所示。

```
public class HelloWorld {
    /**
     *  Title: Hello world
     *  @author lenovo
     *  @Time 2024-08-25 12:45:25
     */
    public static void main(String[ ] args) {
        // TODO Auto-generated method stub
        System.out.println("Hello world!");
    }
}
```

2.4 数据类型

任何一种语言都有自己的数据类型划分，这就对数据在内存中的存储有了严格的限制，不同数据类型保存的数据内容不一样，所占的内存字节数也是不一样的。在 Java 中，总体可以分为基本数据类型（包括 byte、short、int、long、float、double、char 和 boolean）和引用数据类型（包括：类、接口、数组和枚举）。

Java 数据类型划分如图 2-1 所示。

在 Java 这些数据类型中，8 种基本数据类型是 Java 的内嵌类型，在任何操作系统中所占空间都一样，而引用数据类型是有开发人员自定义的数据类型。下面重点来讲解 Java 的

8 种基本数据类型。

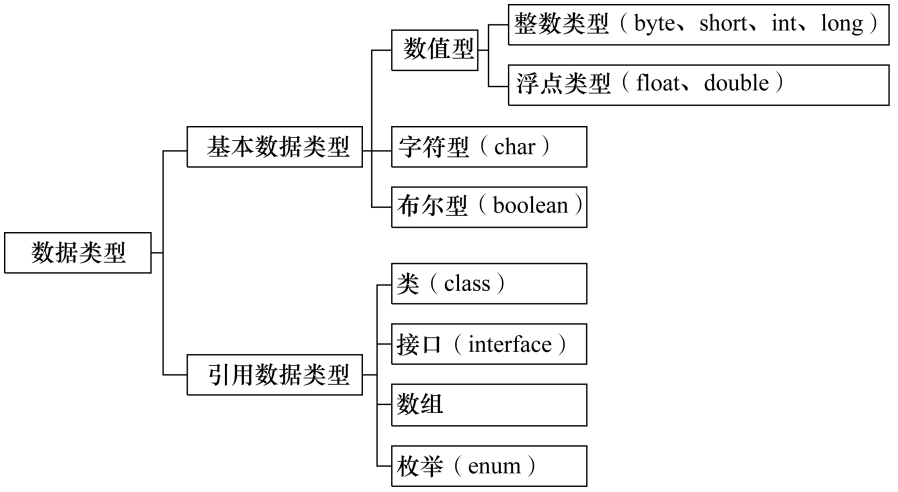


图 2-1　Java 数据类型划分

2.4.1 整数类型

整数类型就是所表示的数据全部为整数，没有小数部分。在 Java 中，又根据存储空间不同分为字节型（byte）、短整型（short）、整型（int）以及长整型（long）这四种类型，它们所占内存空间的大小、数据表示的范围以及默认值如表 2-2 所示。

表 2-2　整数类型

序号	数据类型	大小（字节）	可表示的数据范围	默认值
1	byte（字节）	1	$-2^7 \sim 2^7-1$	0
2	short（短整型）	2	$-2^{15} \sim 2^{15}-1$	0
3	int（整型）	4	$-2^{31} \sim 2^{31}-1$	0
4	long（长整型）	8	$-2^{63} \sim 2^{63}-1$	0

在使用整数类型声明变量并赋值时，除了 long 类型外，其他整数类型的数据直接赋一个整数常量即可；而 long 类型需要在整数常量后面加上一个字母 L（或者小写 l），用于说明这个数据为 long 类型数据，当所赋的值未超出 int 类型的取值范围，可以省略末尾的字母 L（或者小写 l）。具体示例代码如下。

```
byte b=56;
short s=116;
int a=1687;
long n1=112200000L;      //所赋的值超过了 int 类型的取值范围,后面必须加 L
long n2=147;             //所赋的值未超过了 int 类型的取值范围,后面可以不加 L
```

2.4.2 浮点数类型

浮点数类型可以表含有小数的数据。在 Java 中，浮点数类型又细分为单精度型浮点数（float）和双精度型浮点数（double），它们所占内存空间的大小、数据表示的范围以及默认值如表 2-3 所示。

表 2-3　浮点数类型

序号	数据类型	大小（字节）	可表示的数据范围	默认值
5	float（单精度）	4	-3.4×10^{38} ～3.4×10^{38}	0.0
6	double（双精度）	8	-1.7×10^{308} ～1.7×10^{308}	0.0

在表 2-3 中，可以发现 float 类型占 4 个字节的内存空间，与 int 类型占内存空间一样，但 float 类型数据表示的范围要比 int 类型大很多，因为浮点数类型的数据存储格式（浮点类型表示完全按照 IEEE754 标准）与整数类型不一样，它表示的是一个更大的实数范围。

在使用浮点数类型数据赋值时要注意，float 类型的数据要在末尾加字母 F（或者小写 f）；double 类型的数据要在末尾加上字母 D（或者 d），当含有小数的数据不加任何字母时，默认为双精度类型。具体示例代码如下。

```
float f=12.5F;
double s=256.6D;
double=1.7;
```

2.4.3 字符类型

字符类型用关键字 char 表示，它表示的数据为一个字符，占用 2 个字节的内存空间，其默认值为'\u0000'。在表示一个字符数据时，需要使用英文状态下的一对单引号''将字符括起来，例如字符‘A’。具体示例代码如下。

```
char c='A';          //为字符类型变量 c 赋值字符'A'
char ch=98;          //为字符类型变量 c 赋值整型数值 98,它会自动转换成对应的字符'b'
```

2.4.4 布尔类型

布尔类型也可以称为是否类型，用 boolean 表示，该类型的变量值只有 true 和 false 两个值，默认值为 false。具体示例代码如下。

```
boolean b=true;   //为布尔类型变量 b 赋值 true
```

2.5 常量与变量

2.5.1 常量

常量就是在程序中被初始化后不能再改变值的量，不能够再次赋值。例如，整型数字5、浮点数字3.6、字符‘a’等都是常量。在Java中，常量分为整型常量、浮点数常量、字符常量、字符串常量和布尔常量等。

（1）整型常量是整型类型的数据，包括二进制、八进制、十进制、十六进制这四种表示形式，具体如下：

二进制：由数字0和1组成的数字序列，且前面必须以0b或0B开头；

八进制：由数字0～7组成的数字序列，且前面必须以0开头；

十进制：由数字0～9组成的数字序列，是我们最为熟悉的一种数字序列；

十六进制：由数字0～9、字母A～F（字母不区分大小写）组成的序列，且前面必须以0x或0X开头。

整型常量举例如下：

```
0B01001100        //二进制
0367              //八进制
1586              //十进制
0X3A7F            //十六进制
```

（2）浮点数常量是浮点类型的数据，包括单精度型常量和双精度型常量。单精度型常量数据以f或F结尾；双精度型常量数据以d或者D结尾，也可以什么也不写，数据默认为双精度型数据。

浮点型常量举例如下：

```
3.45f             //单精度常量
2.88F             //单精度常量
9.166             //双精度常量
52.35D            //双精度常量
3.67d             //双精度常量
```

（3）字符常量是字符型的数据，字符常量数据是用英文单引号引起来的一个字符。

字符常量举例如下：

```
‘a’               //字符a
‘1’               //字符1
‘?’               //字符?
```

（4）字符串常量是字符串类型的数据，其数据必须用英文双引号引起来的一组字符。字符串常量举例如下：

```
"abc123"        //字符串 abc123
"person"        //字符串 person
```

（5）布尔常量是布尔类型的两个值 true 和 false，逻辑运算及关系运算都会产生一个布尔类型的结果。

2.5.2 变量

在程序运行过程中，会产生很多的临时数据，这些不同类型的数据就会保存在内存制定的单元中，为了存取方便，每个保存数据的内存单元用一个标识符来标识，这些标识符就被称作变量，被标识符标识的内存单元中的数据被称为变量的值。

1. 变量的定义

不同数据类型的变量，在内存中开辟的内存空间是不一样的，所以，在定义变量时，要根据程序的具体需求来为变量开辟内存空间，即不要在程序运行过程中，临时数据出现数据溢出，也不要出现很大内存空间的浪费。

下面通过具体的变量定义进行举例。

```
int a=2;  //定义整型变量 a,并赋值 2,为变量 a 开辟 4 个字节的内存空间
double b=3.5;  //定义双精度型变量 b,并赋值 3.5,为变量 b 开辟 8 个字节的内存空间
```

2. 变量的类型转换

在进行代码开发时，经常会遇到多种不同数据类型的数据进行运算，这时就需要先统一各数据的数据类型。在 Java 中，根据转换方式的不同，数据类型转换分为自动类型转换和强制类型转换。下面就这两种转换方式进行详细讲解。

（1）自动类型转换

自动类型转换也成为隐式转换，是指两种不同数据类型变量进行转换时不需要显式声明。整体的转换规则：一是要满足两种数据类型彼此兼容；二是数据类型表示范围小的可以自动向数据类型表示范围大的自动转换；三是 byte 类型、short 类型与 char 类型之间不会相互转换，它们三者在计算时首先都会转换为 int 类型；四是 boolean 类型不与其他基本数据类型进行转换；五是基本数据类型（包括 boolean 类型）的值与字符串进行连接运算时，基本数据类型自动转换为字符转类型。各数据类型自动转换如图 2-2 所示。

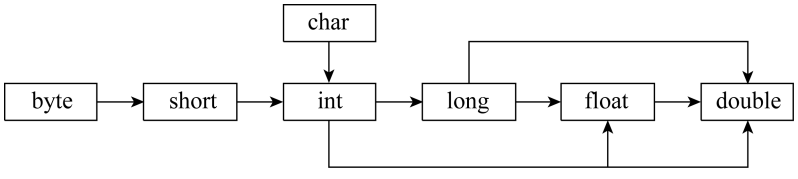


图 2-2　自动类型转换图

下面举例说明自动类型转换过程。

```
int a=5;              //定义整型变量 a,并赋值为 5
double b=2.0, c;      //定义双精度型变量 b、c;并给变量 b 赋值为 2.0
c=b/a;                //计算 b 除以 a,其结果 2.5 赋值给变量 c
```

在上面 b/a 运算过程中，编译器会检测到变量 a 与 b 的数据类型不统一，由于变量 b 的类型为 double，而变量 a 的类型为 int，编译器会将变量 a 的值 5，自动类型转换成 5.0，然后再进行除运算，结果为 2.5。

（2）强制类型转换

强制类型转换也成为显式转换，是指两种不同数据类型变量进行转换时需要显式声明。强制类型转换其实是自动类型转换的逆过程，在数据类型兼容的情况下，将数据类型表示范围大的向数据类型表示范围小的转换时，就需要显式的进行强制类型转换，这个过程可能会数据精度降低或溢出。转换规则是在被强制类型转换的变量、值及表达式前面加一对小括号，在小括号里面输入要强制转换的类型。

下面举例说明强制类型转换过程。

```
int a;                //定义整型变量 a
double b=3.7;         //定义双精度型变量 b,并赋值为 3.7
a=(int)b;             //将变量 b 的值强制类型转换为整型
```

2.6 运算符

在 Java 程序设计中，经常会使用到一些符号，例如+、-、*、/、>、%等，这些符号统称为运算符，通过这些运算符变可以进行相应的运算操作。在这里，可以将运算符按照功能分为算数运算符、赋值运算符、关系运算符、逻辑运算符、位运算符以及条件运算符，下面详细讲解常用几类运算符。

2.6.1 算数运算符

算数运算符主要用于数学运算，不仅包含了加减乘除这样的四则运算符，还有取模等运算。算数运算符的用法如表 2-3 所示。

表 2-3 算数运算符

序号	运算符	运算描述	范例	结果
1	+	加/正号	4+8	12
2	-	减/负号	6-3	3

续表

序号	运算符	运算描述	范例	结果
3	*	乘	3 * 7	21
4	/	除（取整）	7/2	3
5	%	取模（取余）	5/3	2

下面通过一个案例观察算数运算符的使用方法，代码如下。

```
//Demo0201. java
package ch02;
public class Demo0201 {
    public static void main(String[ ] args) {
        // TODO Auto- generated method stub
        int a=10;
        int b=3;
        System. out. println("a+b="+(a+b));
        System. out. println("a- b="+(a- b));
        System. out. println("a * b="+(a * b));
        System. out. println("a/b="+(a/b));
        System. out. println("a%b="+(a%b));
    }
}
```

文件 Demo0201 运行结果如图 2-3 所示。

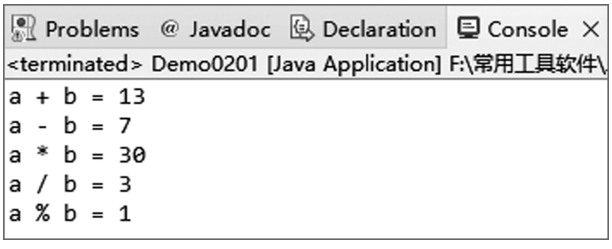


图 2-3　文件 Demo0201 运行结果

从上面程序运行的结果可以观察到，加、减、乘三个运算符与在数学中运算功能一直；但在 Java 中，除运算是取整的含义，无论小数是多少，直接舍去；取模运算，就是取余的运算。

2.6.2　自增/自减运算符

Java 中的自增/自减运算符（++/--）用于将操作数的值增加 1（减少 1）。它可以有

两种形式：前缀形式（++变量/--变量）和后缀形式（变量++/变量--）。这两种形式都会使变量的值增加 1（减少 1），但是它们在表达式中的行为有所不同。自增/自减运算符用法如表 2-4 所示。

表 2-4　自增/自减运算符用法

序号	运算符	运算描述	范例	结果
1	++	自增（前缀）	a=1；b=++a	a=2；b=2
2	++	自增（后缀）	a=1；b=a++	a=2；b=1
3	--	自减（前缀）	a=1；b=--a	a=0；b=0
4	--	自减（后缀）	a=1；b=a--	a=0；b=1

- 前缀形式（++变量/--变量）

在表达式中先增加（减少）变量的值，然后使用新的值，变量的值立即被修改。

- 后缀形式（变量++/变量--）

在表达式中先使用变量的当前值，然后增加（减少）变量的值；表达式的结果是原始值，而变量的值随后被修改。

下面通过一个案例观察运算符的使用方法，代码如下。

```
//Demo0202. java
package ch02;
public class Demo0202 {
    public static void main(String[ ] args) {
        // TODO Auto- generated method stub
        int x=10;
        int y=10;
        int z=15;
        int k=15;
        //使用前缀自增
        int a=++x;                          // x 的值变为 11, a 的值为 11
        System. out. println("++x: "+x);    //输出 11
        System. out. println("a=++x: "+a);  //输出 11
        //使用后缀自增
        int b=y++;                          // y 的值变为 11, 但 b 的值为 10
        System. out. println("y++: "+y);    //输出 11
        System. out. println("b=y++: "+b);  //输出 10
        //使用前缀自减
        int c=- - z;                        // z 的值变为 14, c 的值为 14
```

```
            System. out. println("- - z: "+z);          //输出 14
            System. out. println("c = - - z: "+c);      //输出 14
            //使用后缀自减
            int d = k- - ;                              // k 的值变为 14, 但 d 的值为 15
            System. out. println("k- - : "+k);          //输出 14
            System. out. println("d = k- - : "+d);      //输出 15
        }
    }
```

文件 Demo0202 运行结果如图 2-4 所示。

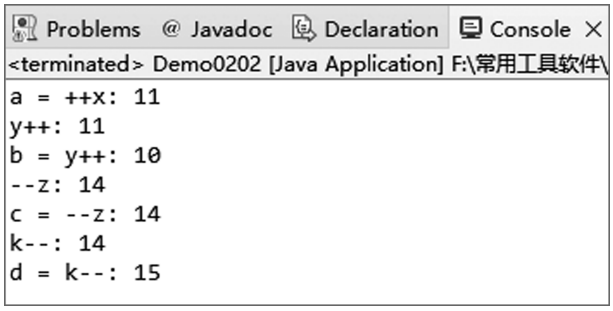


图 2-4　文件 Demo0202 运行结果

2.6.3 赋值运算符

赋值运算符的作用就是将常量、变量等的值赋值给‘=’右侧的另一个变量。下面通过表 2-5 来看一下赋值运算符及其用法。

表 2-5　赋值运算符用法

序号	运算符	运算描述	范例	结果
1	=	赋值	a=5；b=2	a=5；b=2
2	+=	加等于	a=5；b=2；a+=b	a=7；b=2
3	-=	减等于	a=5；b=2；a-=b	a=3；b=2
4	*=	乘等于	a=5；b=2；a*=b	a=10；b=2
5	/=	除等于	a=5；b=2；a/=b	a=2；b=2
6	%=	模等于	a=5；b=2；a%=b	a=1；b=2

在上表中可以观察到，赋值运算符的运算顺序都是从右至左；除了‘=’，其他赋值运算符都是复合的赋值运算符，以‘+=’为例进行说明，a+=b 就相当于 a=a+b。

2.6.4 关系运算符

关系运算符的作用是比较两个常量或者变量的大小，结果是一个布尔类型的值（true或 false），下面通过表 2-6 来说明关系运算符及其用法。

表 2-6 关系运算符用法

序号	运算符	运算描述	范例	结果
1	==	等于	6==5	false
2	!=	不等于	6!=5	true
3	<	小于	6 < 5	false
4	>	大于	6 > 5	true
5	<=	小于等于	6 <=5	false
6	>=	大于等于	6 >=5	true

下面通过一个案例观察运算符的使用方法，代码如下。

```
//Demo0203. java
package ch02;
public class Demo0203 {
    public static void main(String[ ] args) {
        // TODO Auto- generated method stub
        int a=6;
        int b=5;
        System. out. println("a==b: "+(a==b));
        System. out. println("a!=b: "+(a!=b));
        System. out. println("a>b: "+(a>b));
        System. out. println("a<b: "+(a<b));
        System. out. println("a>=b: "+(a>=b));
        System. out. println("a<=b: "+(a<=b));
    }
}
```

文件 Demo0203 运行结果如图 2-5 所示。

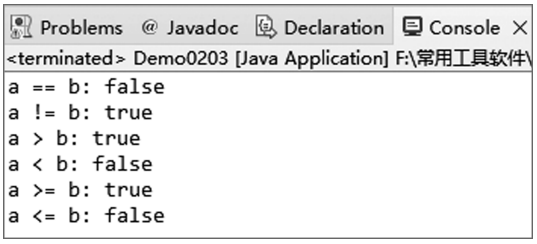


图 2-5　文件 Demo0203 运行结果

2.6.5 逻辑运算符

逻辑运算符的作用是对布尔类型的值或表达式进行运算，其结果是一个布尔类型的值（true 或 false），下面通过表 2-7 来说明关系运算符及其用法。

表 2-7　关系运算符用法

序号	运算符	运算描述	范例	结果
1	&&	逻辑与（短路与）	true && true	true
			true && false	false
			false && true	false
			false && false	false
2	||	逻辑或（短路或）	true && true	true
			true && false	true
			false && true	true
			false && false	false
3	!	非	! true	false
			! false	true

从上表中可以观察到，对于逻辑与，也称为短路与，只要逻辑与运算符左侧的表达式逻辑值为 false，右侧表达式将不再参与运算，被短路；对于逻辑或，也成短路或，只要逻辑或左侧的表达式逻辑值为 true，右侧表达式将不再参与运算，被短路；对于非运算，直接改变成相反的逻辑值。

下面通过一个案例观察运算符的使用方法，代码如下。

```
//Demo0204. java
package ch02;
public class Demo0204 {
    public static void main(String[ ] args) {
```

```
            // TODO Auto-generated method stub
            boolean a=true;
            boolean b=false;
            System.out.println("a && b: "+(a && b));
            System.out.println("a || b: "+(a || b));
            System.out.println("! a: "+!a);
        }
    }
```

文件 Demo0204 运行结果如图 2-6 所示。

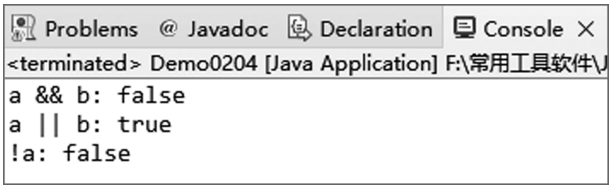


图 2-6　文件 Demo0204 运行结果

2.6.6 条件运算符

条件运算符（?:），也成为三目运算符，它根据一个条件返回两个可能的值之一。其格式如下所示。

（布尔型表达式）？表达式 1 ：表达式 2

当布尔型表达式的值为 true 时，返回表达式 1 的值，否则返回表达式 2 的值。下面通过一个案例来说明运算符的使用方法，代码如下。

```
//Demo0205.java
package ch02;
public class Demo0205 {
    public static void main(String[ ] args) {
        // TODO Auto-generated method stub
        int a=10;
        int b=5;
        int max=(a>b) ? a:b;
        System.out.println("Max:"+max);
    }
}
```

文件 Demo0205 运行结果如图 2-7 所示。

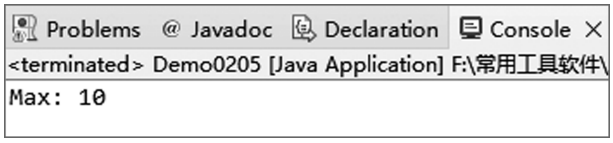


图 2-7　文件 Demo0205 运行结果

2.6.7 位运算符

位运算符是对二进制数据的每一位进行运算的操作符。下面通过表 2-8 来说明位运算符及其用法。

表 2-8　位运算符用法

序号	运算符	运算描述	范例	结果
1	&	按位与	0 & 0	0
			0 & 1	0
			1 & 0	0
			1 & 1	1
2	|	按位或	0 | 0	0
			0 | 1	1
			1 | 0	1
			1 | 1	1
3	~	非	~0	1
			~1	0
4	^	按位异或	0^0	0
			0^1	1
			1^0	1
			1 | 1	0
5	<<	左移	00001000<<2	00100000
			11010001<<2	01000100
6	>>	右移	00101100>>2	00001000
			10110001>>2	11101100
7	>>>	无符号右移	00101100>>>2	00001000
			10110001>>>2	00101100

下面通过一个案例来说明运算符的使用方法，代码如下。

```
//Demo0206. java
package ch02;
public class Demo0206 {
    public static void main(String[ ] args) {
        // TODO Auto- generated method stub
        int a=10; //二进制:1010
        int b=5; //二进制:0101
        System. out. println("a & b(十进制): "+(a & b)); //十进制:0
        System. out. println("a & b(二进制): "+Integer. toBinary
        String(a & b));//二进制:0
        System. out. println("a|b(十进制): "+(a|b));//十进制:15
        System. out. println("a|b(二进制): "+Integer. toBinary
        String(a|b)); //二进制:1111
        System. out. println("a^b(十进制): "+(a^b));
        System. out. println("a^b(二进制): "+Integer. toBinary
        String(a^b)); // 1111
        System. out. println("～a:(十进制) "+～a);//十进制:- 11
        System. out. println("a<<2(十进制): "+(a<<2)); //十进制: 40
        System. out. println("a<<2(二进制): "+Integer. to Binary
        String(a<<2));//二进制:
        System. out. println("a>>2(十进制): "+(a >> 2)); //十进制:2
        System. out. println("a>>2(二进制): "+Integer. toBinary
        String(a >> 2));//二进制:
        System. out. println("a>>>2(十进制): "+(a>>>2)); // 2 (无符号右移)
    }
}
```

文件 Demo0206 运行结果如图 2-8 所示。

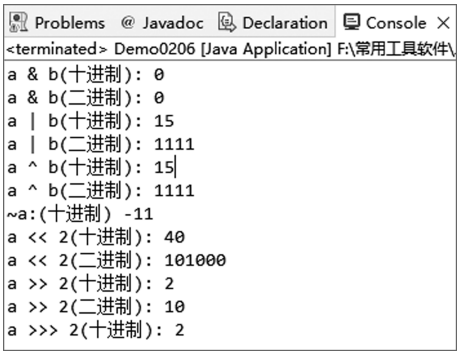


图 2-8　文件 Demo0206 运行结果

需要注意的是，左移操作，二进制左侧高位移走的直接舍去，右侧地位补 0 即可，无需考虑符号位；右移操作（带符号），二进制左侧高位要根据符号位补 0 还是 1，（符号位是负数最高位是 1，就全部补 1；符号位是正数最高位是 0，就全部补 0），右侧移走的部分直接舍去；右移操作（不带符号），无需考虑符号位，直接二进制左侧高位补 0，右侧移走的部分直接舍去。

2.6.8 运算符的优先级

在进行一些复杂的表达式运算时，要根据表达式中各个运算符的优先级顺序进行运算。运算符的优先级如表 2-8 所示。

表 2-9　运算符的优先级

优先级	运算符
1	.　()　[]
2	!　+（正号）　-（负号）　~　++　--
3	*　/　%
4	+、-
5	<<　>>　>>>
6	>　>=　<　<=
7	==　!=
8	&（位运算符 AND）
9	^（位运算符号 XOR）
10	|（位运算符号 OR）
11	&&
12	||
13	?:
14	=　+=　-=　*=　/=　%=

练习题

一、填空题

1. Java 中的标识符组成原则：_____。

2. _____关键字是在 JDK 1.4 时加入的，_____关键字是在 JDK 1.5 时加入的。

3. 列举出已经知道的 5 个关键字：_____ 。

4. Java 注释分为以下三种：_____ 、_____ 、_____。

5. Java 中使用_____ 关键字，可以定义一个整型数据。

6. 在一个 Java 源文件中定义了 3 个类和 15 个方法，编译该 Java 源文件时会产生_____个字节码文件，其扩展名是 _____。

7. 布尔型数据类型的关键字是_____，有_____和_____ 两种取值。

8. 整型数可以采用_____ 、_____、_____和_____四种类型表示。

9. 根据占用内存长度的不同将浮点型分为_____ 和_____ 两种。

10. Java 程序结构分为：_____ 、_____、_____ 三种。

11. 逻辑表达式：true&&false&&true 的结果是_____ 。

12. 逻辑表达式：！true ‖ false 的结果是_____。

13. 在方法中可以使用_____语句来结束方法的执行。

14. 方法中的_____关键字用来表示方法不返回任何值。

二、选择题

1. 下面那些标识符是正确的（　　）。

A. class　　B. hello world

C. 123 $ temp　　D. Demo

2. 下面哪个关键字是 Java 中未使用到的关键字（　　）（多选）。

A. const　　B. class　　C. int　　D. assert

3. public static void main 方法的参数描述是：（　　）。

A. String args[ ]　　B. int[ ]args

C. Strings args[ ]　　D. String args

4. 下面说法正确的是（　　）。

A. Java 程序的源文件名称与主类（public class）名称相同，后缀可以是 . java 或 . txt 等

B. JDK 的编译命令是 java

C. 一个 java 源文件编译后可能产生多个 class 文件

D. 在命令行编译好的字节码文件，只需在命令行直接键入程序名即可运行该程序

5. 下面说法不正确的是（　　）。

A. Java 语言是面向对象的，解释执行的网络编程语言

B. Java 语言具有可移植性，是与平台无关的编程语言

C. Java 语言可对内存垃圾自动收集

D. Java 语言执行时需要 Java 的运行环境

6. 下面（　　）不是 Java 的关键字。

A. integer　　B. double　　C. float　　D. char

7. 在 Java 中，字节数据类型的关键字和默认值是（　　）。

A. byte 和 0　　B. byte 和 1

C. boolean 和 true　　D. boolean 和 false

8. 3. 15E2 表示的数据是（　　）。

A. 3. 15×2　　B. 3. 15×10^{-2}

C. 3. 15×10^{2}　　D. 0. 315×10^{-2}

9. 程序 System. out. println（"1+1="+1+1）；的输出结果是（　　）。

A. 1　　B. 1+1=2　　C. 1+1=11　　D. 2

10. 程序 System. out. println（10 / 3）；的输出结果是（　　）。

A. 1　　B. 3　　C. 3. 3　　D. 3. 33333

11. 执行下面的语句后，a、b、c 的值分别是（　　）。

```
int a=2;
int b=(a++) * 3;
int c=(++a) * 3;
```

A. 2、6、6　　B. 4、9、9　　C. 4、6、12　　D. 3、9、9

12. 以下的（　　）能正确表示 Java 语言中的一个整型常量。

A. 35. d　　B. −20　　C. 1，234　　D. "123"

13. 下面的数据类型（　　）是 float 型

A. 33. 8　　B. 129　　C. 89L　　D. 8. 6F

14. 下列关于自动类型转换的说法中，正确的一个是（　　）。

A. int 类型数据可以自动转换为 char 类型数据

B. char 类型数据可以被自动转换为 int 类型数据

C. boolean 类型数据不可以做自动类型转换，但是可以做强制转换

D. long 类型数据可以被自动转换为 short 类型数据

15. 若有定义 int a=9，b=6；那么 a>b 的值是：（　　）

A. 0　　B. 1　　C. false　　D. true

16. 设 x=1，y=2，z=3，则表达式　y+=z−−/++x　的值是（　　）。

A. 3　　B. 3. 5　　C. 4　　D. 5

17. 下列哪一个不属于 Java 中的 8 种基本数据类型（　　）

A. double　　B. char　　C. int　　D. String

18. 以下数据类型存储空间最大的是（　　）。

A. byte　　B. long　　C. float　　D. char

19. 为一个 boolean 类型变量赋值时，可以使用（　　）方式。

A. boolean=1；

B. boolean a=（9 >=10）；

C. boolean a="真"；

D. boolean a==false；

20. 运算符优先级别排序正确的是（　　）。

A. 由高向低分别是：( )、!、算术运算符、关系运算符、逻辑运算符、赋值运算符；

B. 由高向低分别是：( )、关系运算符、算术运算符、赋值运算符、!、逻辑运算符；

C. 由高向低分别是：( )、算术运算符、逻辑运算符、关系运算符、!、赋值运算符；

D. 由高向低分别是：( )、!、关系运算符、赋值运算符、算术运算符、逻辑运算符；

三、判断题

1. 变量的内容可以修改，常量的内容不可修改。（　　）
2. goto 是 Java 中未使用到的关键字。（　　）
3. enum 关键字是在 JDK 1.4 版本中增加的。（　　）
4. 使用 public class 定义的类，文件名称可以与类名称不一致。（　　）
5. 主方法编写：public void main（String arg）。（　　）
6. 字符 $ 不能作 Java 标识符的第一个字符。（　　）
7. System. out. println( )输出后是不加换行的。（　　）
8. byte 的取值范围是：0～255。（　　）
9. int 和 double 进行加法操作，int 会自动转换为 double 类型。（　　）
10. 使用“&”操作时，如果第一个条件是 false，则后续的条件都不再判断。（　　）
11. 使用“&&”操作时，如果第一个条件是 false，则后续的条件都不再判断。（　　）
12. 使用“|”操作时，如果第一个条件是 true，则后续的条件都不再判断。（　　）
13. 使用“||”操作时，如果第一个条件是 true，则后续的条件都不再判断。（　　）
14. 定义多个同名方法时，可以依靠返回值区别同名方法。（　　）