

# Java 程序设计入门

## 知行启航

本章我们将正式拉开 Java 程序设计的序幕,以技术为舟、以立德树人为帆,共同探索这片充满无限可能的编程海洋。

Java 不仅仅是一门编程语言,它更是一种技术生态,一种思维方式。自 1995 年诞生以来,Java 凭借其“一次编写、到处运行”的跨平台特性,迅速在全球范围内普及开来,成为企业级应用开发、安卓应用开发、大数据处理、云计算等多个领域的首选语言。

然而,技术的学习不应仅仅停留在技能层面。作为新时代的青年学子,我们更应将技术学习与国家发展、社会进步紧密结合,用技术解决实际问题,用编程服务社会,让技术之光照亮前行的道路。

在正式踏入 Java 编程的殿堂之前,编写并运行第一个 Java 程序无疑是一个重要的里程碑。这个过程虽然简单,却蕴含着编程的基本思想和方法,是每一位编程爱好者必须经历的成长仪式。

想象一下,当屏幕上缓缓出现那句“Hello, Java!”时,那份激动与成就感将如同潮水般涌来。这不仅仅是一个简单的字符串输出,更是你与计算机世界建立联系的桥梁,是你编程梦想启航的起点。



## 1.1 Java 概述



1.1 Java 概述

### 1.1.1 案例 1-1: 时光穿梭者的 Java 之旅

李明,一位对历史满怀好奇的软件工程师,偶得一“时光穿梭机”,可穿梭时空,亲历科技变迁。他首探 1995 年 Sun Microsystems(简称 Sun)公司,目睹 Java 横空出世,亲历“一次编写,到处运行”理念的诞生,以实习生身份参与 Java 1.0 测试,与 Java 之父 James Gosling 等交流,洞悉 Java 诞生之初的蓝图与挑战。

继而,李明来到 2005 年,正值 Java 企业应用辉煌时期,他加入金融科技企业,亲历 J2EE 大型系统开发,见证 Spring 框架兴起及 JSP、Servlet 重塑 Web 开发。



最终,李明回归当下,并窥视未来 Java 图景,包括大数据(Hadoop、Spark)、云计算(Docker、Kubernetes 部署)及微服务架构中的广泛应用,同时探索 Java 在 AI、物联网等领域的创新前沿。

### 1.1.2 知识预备站

#### 1. Java 语言的诞生

Java 源自 C++,是 Sun 公司(后被 Oracle 公司收购)的工程师 James Gosling 等在 20 世纪 90 年代初共同打造的。其前身 Oak 语言在 1991 年开始研发。1995 年,在 Oak 更名为 Java 后,Java 在 Sun World 会议上正式诞生,标志着这一革命性编程语言的问世。1996 年 1 月,Sun 公司发布了第一个 JDK(Java Development Kit)版本——JDK 1.0,为 Java 开发者提供了基本的开发工具和平台。

随着 Java 技术的不断成熟,越来越多的操作系统供应商开始在其产品中嵌入 Java 技术。1996 年 4 月,10 个最主要的操作系统供应商宣布了这一决定。2006 年,Sun 公司宣布将 Java 技术作为免费开源软件对外发布,并成立了 OpenJDK 开源社区组织,对 Java 源码进行独立管理。这一举措极大地推动了 Java 的发展,吸引了全球大量的开发者和组织加入 Java 社区。2009 年,甲骨文公司宣布收购 Sun Microsystems,成为 Java 技术的新主人。此后,甲骨文继续推动 Java 的发展,并发布了多个新版本。每个版本都引入了新的特性和改进,旨在提升 Java 的性能、安全性和易用性。具体可以在甲骨文的官方网站中进行查阅。

#### 2. Java 语言的特点

Java 语言作为一种广泛使用的编程语言,具有多个显著的特点,这些特点使得 Java 成为企业级开发、移动应用开发、大数据处理以及云计算等多个领域的首选语言。以下是 Java 语言的一些主要特点。

(1) 跨平台性。Java 语言最大的特点之一是一次编写、到处运行。这是通过 Java 虚拟机(JVM)实现的,JVM 可以在任何支持它的操作系统上运行 Java 程序,从而实现了 Java 程序的跨平台性。

(2) 面向对象。Java 是一种纯面向对象的编程语言,它支持封装、继承和多态等面向对象的基本特性。Java 中的一切都可以视为对象,包括基本数据类型(通过包装类实现)。

(3) 自动垃圾收集。Java 提供了自动垃圾收集机制,用于自动回收不再被使用的对象所占用的内存空间,从而简化了内存管理,避免了内存泄漏和内存溢出等问题。

(4) 丰富的库。Java 拥有庞大的标准库,提供了大量的预定义的方法和类,用于执行常见的任务,如输入/输出、网络编程、日期时间处理等。此外,还有专门为企业级应用(Java Enterprise Edition、Java EE)和移动设备(Java Mobile Edition,现已被 Android 取代)设计的库。

(5) 安全性。Java 设计时就考虑到了安全性,提供了许多内置的安全特性,如访问控制、加密、解密、数字签名等。Java 的自动内存管理和沙箱模型(Sandbox Model)也有助于防止恶意代码的执行。

(6) 健壮性。Java 在编译和运行时都会进行严格的类型检查,这有助于在编译阶段就发现潜在的错误,提高程序的健壮性。此外,Java 还提供了异常处理机制来捕获和处理运行时错误。

(7) 多线程。Java 支持多线程编程,允许多个任务同时执行。Java 提供了丰富的线程



控制和管理工具,使得编写高效的多线程程序变得相对容易。

(8) 泛型。从 Java 5 开始,Java 引入了泛型编程的概念,允许在编译时检查集合中元素的类型,避免了类型转换的错误,提高了代码的可读性和安全性。

(9) 注解。注解是 Java 5 引入的一种语法元素,它为代码提供了一种形式化的元数据信息。注解可以用于生成文档、进行编译时检查以及运行时处理。

(10) 易于学习和使用。尽管 Java 拥有强大的功能和复杂的特性,但其语法清晰、简洁,且拥有大量的学习资源和社区支持,使得 Java 相对容易学习和使用。

### 3. Java 的应用领域

Java 作为一种广泛使用的编程语言,其应用领域非常广泛。以下是 Java 的几个主要应用领域。

(1) 坚实基础。Java 凭跨平台、高稳定性和社区支持,深入互联网应用。其工具和框架(如 Spring Boot、Hibernate)降低了开发门槛,促进了创新与快速迭代。

(2) Web 核心。Java 作为 Web 服务器与框架基石,支持高性能、可扩展 Web 应用开发,满足多样需求,确保应用稳健运行。

(3) 微服务引领者。Java 通过 Spring Cloud 等推动微服务架构,提升系统灵活性与可维护性,加速部署与扩展,为互联网应用注入新动力。

(4) 大数据先锋。Hadoop、Spark 等大数据框架基于 Java,提供高效可靠的数据处理能力,简化大数据处理流程。

(5) 数据存储强援。Java 广泛兼容,与各类数据库无缝对接,内置丰富数据结构与算法库,助力复杂数据处理。

(6) 实时数据处理利器。Java 多线程与并发编程优势,确保高并发数据处理时效性与准确性,满足实时数据处理需求。

(7) 机器学习赋能者。Java 在 AI 领域很重要,其机器学习库支持算法与模型,简化构建与训练,融合大数据、云计算技术,加速 AI 应用落地。

(8) NLP 助手。Java 借 Stanford NLP、Apache OpenNLP 等工具,轻松处理语义分析、文本挖掘,为 NLP 应用开发奠基。

(9) 跨平台安全保障。Java 跨平台运行,拓宽 AI 应用范围;其安全机制保护数据安全与隐私,让用户安心享受智能技术。

综上所述,Java 应用在网站开发、移动应用开发、企业级应用开发、大数据处理、云计算与微服务、嵌入式系统开发、人工智能游戏开发、科学计算、交互式系统开发等多个领域。Java 的通用性和高效性使得它成为这些领域的重要编程语言之一。

### 4. Java 开发环境的搭建

Java JDK 是 Java 语言的软件开发工具包,主要由以下几部分组成。

(1) Java 开发工具。提供了 Java 开发所需的各种工具,如编译器、调试器、重构工具等。这些工具帮助开发者编写、调试和优化 Java 代码。

(2) Java 平台标准类库。提供了 Java 应用程序所需的类库和接口,包括输入/输出、网络通信、多线程等功能。这些类库和接口使得开发者能够更方便地实现各种功能。

(3) Java 虚拟机。JVM 是 Java 运行环境的核心,负责执行 Java 字节码。JVM 使得 Java 应用程序能够在任何安装了 JVM 的平台上运行,而不需要重新编译,从而实现了 Java 的跨平台性。



Java JDK 的安装是一个相对直接的过程,但具体步骤可能会因操作系统(如 Windows、macOS、Linux)和 JDK 版本的不同而有所差异。以下是一个通用的安装流程,以 Windows 系统为例,同时也涵盖了 macOS 和 Linux 系统的简要说明。

#### 1) Windows 系统安装步骤

- (1) 确定操作系统和 JDK 版本。
- (2) 确定用户的计算机操作系统是 32 位还是 64 位。
- (3) 访问 Oracle 官网(Oracle 官网)。
- (4) 根据用户的操作系统和需要,下载合适的 JDK 版本安装包。

#### 2) 下载 JDK 安装包

在 Oracle 官网的 Java SE 下载页面上,如图 1-1 所示,单击“立即下载 Java”按钮,可进入下载页面。如图 1-2 所示,选择适合用户操作系统的 JDK 版本(如 Windows x64 Installer)进行下载。

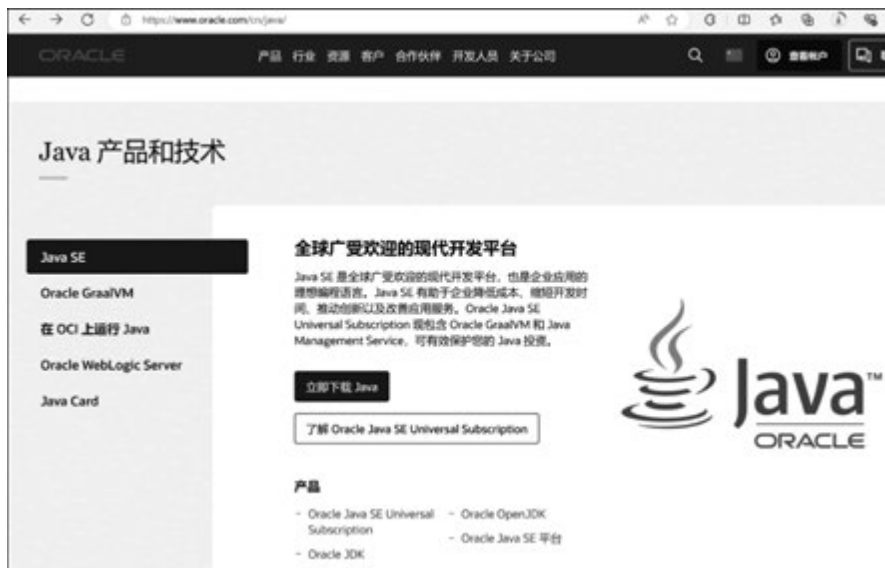


图 1-1 Java 官网界面

JDK 22	JDK 21	JDK 17	GraalVM for JDK 22	GraalVM for JDK 21	GraalVM for JDK 17
<strong>JDK Development Kit 22.0.2 downloads</strong>					
JDK 22 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions (NFTC).					
JDK 22 will receive updates under these terms, until September 2024, when it will be superseded by JDK 23.					
<strong>Linux   macOS   Windows</strong>					
Product/File description	File size	Download			
ARM64 Compressed Archive	184.27 MB	<a href="https://download.oracle.com/java/22/latest/jdk-22_linux-aarch64_bin.tar.gz">https://download.oracle.com/java/22/latest/jdk-22_linux-aarch64_bin.tar.gz</a> (sha256)			
ARM64 RPM Package	183.95 MB	<a href="https://download.oracle.com/java/22/latest/jdk-22_linux-aarch64_bin.rpm">https://download.oracle.com/java/22/latest/jdk-22_linux-aarch64_bin.rpm</a> (sha256) (OL 8 GPG Key)			
x64 Compressed Archive	186.23 MB	<a href="https://download.oracle.com/java/22/latest/jdk-22_linux-x64_bin.tar.gz">https://download.oracle.com/java/22/latest/jdk-22_linux-x64_bin.tar.gz</a> (sha256)			
x64 Debian Package	159.64 MB	<a href="https://download.oracle.com/java/22/latest/jdk-22_linux-x64_bin.deb">https://download.oracle.com/java/22/latest/jdk-22_linux-x64_bin.deb</a> (sha256)			
x64 RPM Package	185.89 MB	<a href="https://download.oracle.com/java/22/latest/jdk-22_linux-x64_bin.rpm">https://download.oracle.com/java/22/latest/jdk-22_linux-x64_bin.rpm</a> (sha256) (OL 8 GPG Key)			

图 1-2 JDK 下载



### 3) 运行安装程序

(1) 双击下载的 JDK 安装包文件,启动安装向导。

(2) 遵循安装向导的指示进行操作,包括接受许可协议、选择安装目录等。

**注意:** 在安装过程中,可以选择是否安装公共 JRE。如果只是作为开发人员使用 JDK,通常不需要额外安装 JRE。

### 4) 配置环境变量(可选,但推荐)

(1) 依次打开“控制面板”→“系统和安全”→“系统”→“高级系统设置”→“环境变量”。

(2) 在系统变量区域中,选择“新建”按钮来创建 JAVA\_HOME 环境变量,并将其值设置为 JDK 的安装目录(例如 C:\Program Files\Java\jdk-XX,其中 XX 是 JDK 的版本号)。

(3) 在系统变量中找到 Path 变量,并编辑它以包含 %JAVA\_HOME%\bin。这样,就可以在命令行中直接运行 Java 和 javac 等命令了。

### 5) 验证安装

(1) 打开命令提示符(cmd)。

(2) 输入 java -version 并按 Enter 键。如果安装成功,将显示 Java 的版本信息。

(3) 输入 javac -version 并按 Enter 键。如果安装成功,将显示 Java 编译器的版本信息。

## 1.1.3 案例参考源码及分析思考

```
//Hello.java
1 public class Hello {
2     public static void main(String[] args) {
3         System.out.println("Hello!");
4     }
5 }
```

### 【源码分析】

(1) 这是 Java 编程中基础也比较经典的示例,用于在屏幕上输出 Hello! 消息。使用计算机的“记事本”书写以上程序,并保存成 java 扩展名(注意,代码最左边的数字 1、2、3、4、5 是行号,不用输入,本书中其他章节中的源码也是如此)。记事本是最基本的文本编辑器之一,它无须安装,几乎所有操作系统都预装了它(或类似的文本编辑器,如 Linux 的 gedit 或 Mac 的 TextEdit)。这使得初学者可以立即开始编写代码,不必担心复杂的安装过程或学习新软件的界面。所以对于初学者来说,在掌握编程基础时,使用记事本仍然是一个很好的起点。

(2) Java 程序的文件扩展名是 .java。这个扩展名在 Java 编程中具有特殊的含义和用途,它是 Java 源代码文件的标识。Java 源代码是程序员用 Java 语言编写的文本文件,包含了程序的源代码和类的定义。这些文件通过 Java 编译器(如 javac)进行编译,生成扩展名为 .class 的字节码文件,这些字节码文件可以在 Java 虚拟机(JVM)上执行。

(3) 使用 javac 命令编译 Java 文件。javac 是 Java 编译器,它会将 .java 文件编译成 .class 文件,这是 Java 字节码文件,可以在 JVM 上运行。如果编译成功,不会有任何输出,但会看到在同一目录下生成了一个名为 Hello.class 的文件。最后使用 java 命令来运行



Java 程序。注意,这里不需要包含 .class 扩展名。运行结果为输出了 Hello!,如图 1-3 所示。



图 1-3 命令行模式下运行 Java 程序

**【思考】** 从官方网站(如 Oracle 官网)下载 JDK 安装包进行安装,并开始编写运行第一个程序。在这一过程中,可能会出现各种各样不顺利的情况。例如,下载速度缓慢或中断,安装过程中遇到权限问题,环境变量配置错误,或者编写代码时语法错误频出。面对这些挑战,初学者需要保持耐心,逐步排查问题。经过一番努力,当看到控制台输出 Hello! 时,那种成就感会激励继续在编程的道路上探索前行。每一个看似不起眼的小问题都是成长路上宝贵的经验积累。不断实践,不断学习,将更加熟练地掌握 Java 编程,开启精彩的编程之旅。

## 1.2 面向对象程序设计



1.2 面向对象程序设计

### 1.2.1 案例 1-2: 小明与面向对象编程的奇妙之旅

小明是一名刚接触编程的高中生,他对计算机科学充满了好奇。在学校的 Java 程序设计课程中,老师决定从面向对象编程(OOP)开始讲起。为了帮助小明和其他同学更好地理解这一概念,老师决定用一个有趣的故事引导他们入门。

小明收到了一个特殊的礼物——一个名为编程魔盒的玩具。这个魔盒有着神奇的力量——能够将小明的想法变成现实。他打开编程魔盒,发现里面有一个名为编程世界的地图。地图上标注了许多神秘的地方,如“类王国”“对象村庄”和“方法森林”。

在类王国,小明遇到了一个名叫类大师的智者。类大师说:“小明,想象一下,你要建造一个房子,首先需要设计房子的蓝图,这个蓝图就是类。然后,你可以根据蓝图建造出许多相同的房子,这些房子就是对象。”

小明又来到了对象村庄,村长说:“小明,每个对象都是独一无二的,但它们都遵循相同的规则。就像我们村庄的村民,虽然性格各异,但都遵循着村庄的规章制度。”

在方法森林,小明学会了如何编写方法,这些方法就像是对象的技能。小明发现,通过



方法,他可以让对象执行特定的动作,如计算、移动或者改变状态。

通过这次奇妙之旅,小明掌握了面向对象编程的基本魔法。他回到了现实世界,充满信心地开始了 Java 程序设计的学习。小明明白了,面向对象编程不仅是一种技术,更是一种思考问题的方式。而这,只是他编程旅程的起点。

## 1.2.2 知识预备站

### 1. 程序设计语言发展历程

程序设计语言的发展过程是计算机科学历史上的一个重要部分,它伴随着计算机硬件和软件的进步而不断发展。最早的计算机程序是直接用机器语言编写的,它由一系列二进制代码组成,这些代码直接对应于计算机的硬件指令。为了简化编程,人们开发了汇编语言,它使用助记符代表机器指令,使得编程更加容易理解和记忆。第一个高级编程语言是 FORTRAN(1955 年),随后出现了 ALGOL(1958 年)、Lisp(1958 年)和 COBOL(1959 年)。这些语言提供了更接近人类语言的语法,使得编程更加抽象。结构化编程语言如 C(1972 年)和 Pascal(1970 年)强调了代码的结构化和模块化,减少了 goto 语句的使用,提高了代码的可读性和可维护性。脚本语言如 Perl(1987 年)、Python(1991 年)和 Ruby(1995 年)等提供了快速开发和易于学习的特点。这些语言通常是解释型语言,不需要编译即可运行。随着多核处理器和分布式系统的普及,出现了如 Erlang(1986 年)、Go(2009 年)等支持并行和分布式计算的语言。随着互联网的普及,出现了用于 Web 开发的编程语言和框架,如 JavaScript(1995 年)、Java(1995 年)、PHP(1995 年)、Ruby on Rails(2005 年)等。

人工智能语言作为未来发展的一个趋势,逐渐崭露头角。这类语言将更加注重自然语言的处理、知识的表示和推理等方面的能力,以更好地支持人工智能应用的开发。程序设计语言的发展经历了从低级到高级、从具体到抽象的不断演进过程。随着计算机技术的不断发展和应用需求的不断变化,未来的程序设计语言将更加智能化、人性化和易用化。

### 2. 面向对象编程思想

面向对象编程是一种编程范式,它将数据和操作数据的方法组织在一起,形成所谓的对象。OOP 的核心思想是将现实世界中的事物抽象为程序中的对象,并通过对象之间的交互完成程序的功能。面向对象编程有以下几个主要概念和思想。

(1) 对象是面向对象编程的基本单位,它是一个包含数据和行为的软件构件。在现实世界中,对象可以是任何实体,如人、汽车、账单等。

(2) 类是对象的蓝图或模板。它定义了一组具有相同属性(数据)和方法(行为)的对象。通过类,可以创建具有相同特征和行为的多个对象。

(3) 封装是将对象的实现细节隐藏起来,只暴露出有限的接口与外界进行交互。这有助于保护对象的数据不被外界直接访问,保证数据的安全性和一致性。

(4) 继承允许创建一个新的类(子类)来继承一个现有类(父类)的属性和方法。这样,子类可以重用父类的代码,并且可以添加新的属性和方法或覆盖(重写)父类的方法。

(5) 多态指的是同一个行为具有多个不同表现形式或形态的能力。在 OOP 中,多态通常通过继承和接口实现,允许不同类的对象对同一消息做出响应。

(6) 抽象是将复杂系统的细节隐藏起来,只展示出必要的部分。在面向对象编程中,抽象可以通过抽象类和接口实现,它们定义了类的结构和行为,但不提供具体的实现。





面向对象编程已经成为软件开发中广泛采用的一种范式,许多流行的编程语言如 Java、C++、Python、C# 等都支持面向对象编程。

### 1.2.3 案例参考源码及分析思考

```
//Circle.java
1 public class Circle {
2     double r;
3     public double getArea()
4     {return 3.14*r*r;}
5 }
6 }
```

#### 【源码分析】

(1) 行 1 的代码定义了一个名为 Circle 的公共类。在 Java 中,每个源文件通常包含一个公共类,且类名应与文件名相同,即这里的 Circle.java。

(2) 行 2 的代码声明了一个类型为 double 的成员变量 r,它用来存储圆的半径。double 是一种数据类型,用于表示双精度浮点数。

(3) 行 3 代码定义了一个名为 getArea 的公共方法,它没有参数,并返回一个 double 类型的值。public 关键字表示这个方法可以被任何其他类访问。

(4) 行 4 代码是 getArea 方法的方法体,它包含了一个返回语句。return 关键字用于从方法中返回一个值。这里,方法计算圆的面积并返回结果,计算公式为  $\pi * r^2$ ,其中  $\pi$ (圆周率)在这里用 3.14 近似表示。

(5) 上述代码只是纯粹定义了一个“圆”的类,其本身不包含 main 方法,这意味着它没有一个入口点来执行。可以直接编译它,但是无法直接运行它,因为它没有提供任何可执行的代码。

**【思考】** 初学者学习面向 OOP,就像打开了一扇通往新世界的大门。在此之前,可能对编程的了解仅限于一些基础逻辑和语法。当开始接触 OOP 时,会感受到一种前所未有的思维方式的转变,万物皆对象是一个核心概念,它体现了 OOP 的基本哲学,对象是具有属性(数据)和方法(行为)的实体,对象是现实世界实体的抽象表示,它可以是一个人、一个地点、一个事物或者一个概念。这种思维方式有助于创建更加清晰、可维护、可扩展的代码。通过将现实世界的问题转换为对象模型,OOP 使得软件开发更加接近人类解决问题的自然方式。

## 1.3 第一个 Java 程序



1.3 第一个 Java 程序

### 1.3.1 案例 1-3: Hello Java

新入职的 Java 开发工程师李明迎来了他的第一项挑战:编写一个简单的 Java 程序,该程序能在控制台或命令行界面上输出 Hello Java。

李明首先细致阅读了任务要求,明确了程序的核心功能——在运行时输出标准的





Hello Java 文本,同时确定了程序需稳定运行、输出纯净无杂质,且需在标准 Java 开发环境(如 JDK)中执行的非功能性需求。

随后,他深入分析了可能遇到的挑战,包括 JDK 的安装与版本兼容性问题、编码过程中的语法或逻辑错误,以及输出格式的精确控制。针对这些潜在问题,李明制定了预防措施:确保 JDK 版本适配并配置好环境,参考官方资源避免编码错误,并通过自审和测试确保程序在各种环境下均能正确执行。

一切准备就绪后,李明开始打开集成开发工具 Eclipse 动手编程。他遵循 Java 的编程规范,构建了一个包含 main 方法的类,并精确无误地编写了输出 Hello Java 的语句。经过编译与运行,程序在控制台上展现了预期的输出。

最后,李明将整个过程、成果及所遇问题的解决方案整理成文档,与项目经理进行了汇报与沟通。项目经理对李明的工作给予了高度评价,并提供了进一步的指导,鼓励他在未来的项目中继续展现这样的专业素养与细致态度。

### 1.3.2 知识预备站

#### 1. 最简单的 Java 程序结构

一个简单的输出一行代码的 Java 程序结构通常遵循以下规则。

(1) 类定义。Java 程序由类组成。每个 Java 文件通常包含一个公共类(public class),其名称与文件名相同(包括大小写)。

(2) 主方法。每个 Java 应用程序都必须有一个 main 方法,它是程序的入口点。main 方法必须被声明为 public static void,并且接收一个字符串数组作为参数。

(3) 输出语句。使用 System.out.println 或 System.out.print 方法输出文本到控制台。println 会在输出后添加一个新行,而 print 则不会。

#### 2. 集成化开发环境工具 Eclipse 下载安装

Eclipse 是一款功能强大的开放源代码集成开发环境(IDE),它主要用于 Java 开发,但同时也支持其他多种编程语言,如 C/C++、PHP、Python 等。Eclipse 凭借其可扩展性、跨平台性和灵活性,在软件开发领域得到了广泛应用。

访问 Eclipse 官方网站 <https://www.eclipse.org/downloads/>,选择如图 1-4 所示的 Download x86-64 按钮进入下载页面,按照提示进行下载。也可以选择下方的超链接



图 1-4 下载页面 1



Download Packages 进入下载页面,如图 1-5 和图 1-6 所示,用户可根据自己的计算机配置选择相应的文件进行下载。对于初学者,只是进行 Java 开发,那么下载 Eclipse IDE for Java Developers 即可。

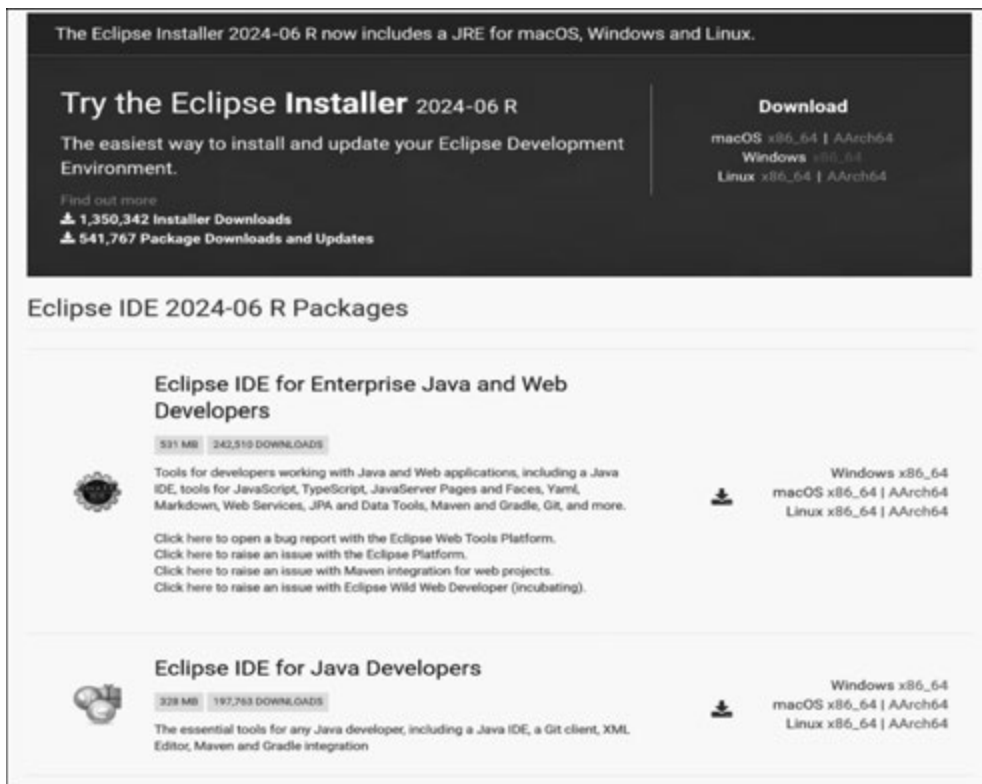


图 1-5 下载页面 2

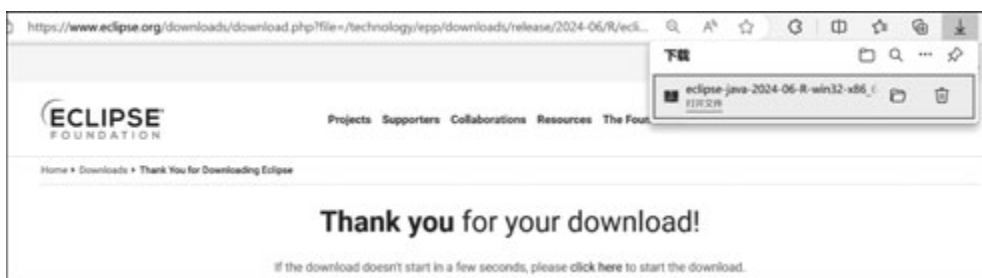


图 1-6 下载成功页面

双击下载好的 eclipse-java-2024-06-R-win32-x86\_64 压缩包进行解压缩,得到如图 1-7 所示的文件夹解压缩界面,双击 eclipse 图标,按照提示操作,即可打开 Eclipse 软件,如图 1-8 所示。

### 3. 使用 Eclipse 开发一个 Java 程序

安装完成后,启动 Eclipse。如果是首次启动,可能会遇到工作空间(Workspace)选择对话框,如图 1-9 所示。工作空间是 Eclipse 存储项目文件的目录,用户可以选择一个现有的