第 1 章　开启 Java 之旅——初识 Java

学习目标

- 了解 Java 语言的诞生背景与发展轨迹。
- 理解 Java 的主要特点与实现机制。
- 掌握 JDK 的下载和安装。
- 掌握 IntelliJ IDEA 的基本使用。
- 掌握开发 Java 程序的步骤。
- 了解项目开发需求分析的内容。
- 厚植家国情怀，拓展全球视野，培养新时代所需的综合素养。

拓展阅读

1.1　Java 语言简介

1.1.1　Java 语言的诞生和发展

1.1.1　Java 语言的诞生和发展

Java 语言由詹姆斯·高斯林(James Gosling)等人研发，其诞生与 SUN 公司的“绿色计划”(green project)紧密相关。20 世纪 90 年代初，为解决消费电子设备(如智能家电)之间的通信与交互难题，高斯林团队开发出 Java 语言雏形，当时它被命名为 Oak。20 世纪 90 年代中期，互联网飞速发展，Sun 公司敏锐察觉到 Oak 在互联网领域的应用潜力，于是对其优化改进，适配网络环境下的开发需求，将名称改为 Java，并于 1995 年 5 月正式推向市场。

1. Java 发展历程

在 Java 正式发布后，其发展历程持续推进，各版本不断迭代升级，逐步构建起完善的技术生态——以下便是 Java 从 1.0 到 21 版本的关键发展里程碑。

(1) Java 1.0(1995 年)：首次正式发布，整合 Java 语法、虚拟机(JVM)与核心类库。凭借“一次编写、处处运行”的跨平台特性奠定生态基础，吸引开发者广泛关注。

(2) Java 1.2(1998 年)：推动企业级开发，引入 JavaServlet、JSP、EJB 等服务端技术，使 Java 成为主流企业应用开发语言，支撑大型后端系统建设。

(3) Java 2(1998 年)：体系化升级，新增 GUI/网络/IO/数据结构等类库；分化为 JavaSE(标准版)、JavaEE(企业版)、JavaME(小型版)，满足多端开发需求。

(4) Java 5(2004 年)：引入泛型、枚举、注解、并发包(java.util.concurrent)，显著提升类

型安全、代码简洁度与并发处理能力,实现了语言层革命。

(5) Java 6(2006 年):优化迭代期,增强 JVM 性能与安全性,完善 API 及开发工具,提升企业系统稳定性与运行效率。

(6) Java 7(2011 年):开发体验升级,支持 switch 处理字符串、try-with-resources 自动资源管理;持续优化 JVM 性能与工具链。

(7) Java 8(2014 年):现代 Java 起点,引入 Lambda 表达式、Stream API 支持函数式编程;重构日期时间库(java.time),革新集合操作与异步流处理。

(8) Java 9(2017 年):实现架构级进化。推出模块化系统(JPMS)解决依赖管理难题;增加语法糖与工具链优化,提升大型应用可维护性。

(9) Java 10～13(2018—2019 年):敏捷迭代期,聚焦局部变量类型推断(var)、GC 优化、JVM 底层性能强化,持续简化编码并提升运行效能。

(10) Java 17(2021 年):LTS 基石版本。引入密封类(sealed classes)精细控制继承关系、模式匹配 switch 简化类型判断;预览虚拟线程(Project Loom 雏形),推进高并发新生态。

(11) Java 21(2023 年):实现并发能力跃迁。正式发布虚拟线程(virtual threads)支持百万级轻量并发;结构化并发统一异步任务管理;分代 ZGC 提升吞吐与低延迟表现。之后,JDK 版本进入 6 个月作为一发布周期,不断迭代更新。最新版本持续推进语言和平台发展。

2. Java 技术的三个主要平台

为满足不同开发领域的需求,Java 技术被划分为三个主要平台:JavaSE、JavaEE 和 JavaME。

- JavaSE(standard edition,标准版):为开发桌面应用程序和商业应用提供核心解决方案。它包含标准的 Java 虚拟机(JVM)和基础类库(如集合、I/O、数据库连接、网络编程等)。JavaSE 是三个平台的核心基础,JavaEE 和 JavaME 均构建在其上。
- JavaEE(enterprise edition,企业版):在 JavaSE 的基础上扩展,专为企业级应用程序开发设计。它提供了丰富的 API 和库(主要包括 Servlet、JSP、JavaBean、JDBC、EJB、Web Services 等技术),用于开发、组装和部署大规模、分布式、高可靠性的企业应用。
- JavaME(micro edition,微型版):面向消费电子产品和嵌入式设备上的应用程序开发。它为资源受限(如内存、处理能力有限)的小型数字设备提供了定制的 Java 运行环境和库。

1.1.2 Java 语言的特点

Java 作为一种面向对象语言,具有自己鲜明的特点,包括简单、面向对象、可移植性、安全性、多线程、健壮性、分布式、体系结构中立、解释执行、高效能、动态等特性,因此日益成为图形用户界面设计、Web 应用、分布式网络应用等软件开发中方便高效的工具。

1. 跨平台性

Java 最核心的机制是“一次编写,到处运行”。这得益于它将源代码编译成与特定硬件和操作系统无关的字节码,这些字节码由安装在目标平台上的 Java 虚拟机(JVM)来执行。开发者只需编写一次程序,就能在支持 JVM 的各种操作系统(如 Windows、Linux、macOS

等)上运行,极大地简化了软件的部署和维护。

2. 面向对象

Java 是一门设计纯粹、贯彻始终的面向对象语言(基本数据类型除外)。它要求程序围绕对象来构建,并严格遵循封装(隐藏内部细节)、继承(建立层次关系并复用代码)、多态(同一接口不同实现)和抽象(定义核心契约)这四大基本原则。这种范式使代码结构清晰、模块化程度高,显著提升了可复用性、可维护性和扩展性。

3. 简单性

Java 的设计目标之一就是易于学习和使用。它借鉴了 C/C++ 的语法使其对许多开发者熟悉,但刻意摒弃了其中复杂、易错和晦涩的特性,如显式的指针操作、运算符重载和多重继承。其自动垃圾回收机制免去了程序员手动管理内存的烦琐任务和潜在错误,同时,庞大且功能完善的标准库为常见开发需求提供了现成的解决方案,降低了开发复杂度和入门门槛。

4. 安全性

安全性是 Java 体系结构中的重中之重。它构建了多层防护机制:字节码在加载时需经过严格验证,确保其合法且无恶意操作;访问控制修饰符(public/private 等)界定类、方法、变量访问范围,防未经授权操作;摒弃 C/C++ 指针,用引用结合自动内存管理(垃圾回收),减少内存风险;此外,标准库内置了强大的加密、认证和安全通信 API,为构建安全应用提供了坚实基础。

5. 多线程支持

Java 在语言层面和核心库中内建了强大且相对易用的多线程能力。开发者可以直接使用类和接口创建线程,也可以通过高级并发工具(如线程池、锁、信号量、并发集合等)高效地构建并发程序。这使开发者能够充分利用现代多核处理器的计算能力,提升程序的响应速度、吞吐量和整体性能。

6. 分布式支持

Java 拥有强大的网络与分布式计算能力。核心库(如 java.net)提供 Socket 通信、URL 处理等基础网络 API;原生支持 RMI 简化跨 JVM 对象通信,是分布式系统核心;同时完善支持 WebServices(JAX-WS、JAX-RS)等现代架构,成为企业级分布式应用开发的首选。

1.1.3　Java 语言的工作机制

Java 之所以广泛应用于各类平台和系统开发,离不开其独特的编译和运行机制。本小节将系统介绍 Java 程序从编写、编译到执行整个流程背后的关键机制,特别是 JVM(Java 虚拟机)和字节码在其中所起的基础性作用。如图 1-1 所示,Java 语言的工作机制分为编译阶段和运行阶段。

1. 编译阶段:生成平台无关的字节码

开发者编写的.java 源代码文件需经过 Java 编译器的编译处理。这一过程并非直接生成特定平台的机器码,而是将源代码转换为一种中间形式——以.class 扩展名的字节码文件。字节码采用了与操作系统和硬件架构无关的统一格式,它包含了 Java 程序的所有指令和元数据,但不针对任何特定平台。这种设计使得同一份字节码文件可以在任何安装了 Java 虚拟机(JVM)的环境中运行,为 Java 的跨平台特性奠定了基础。

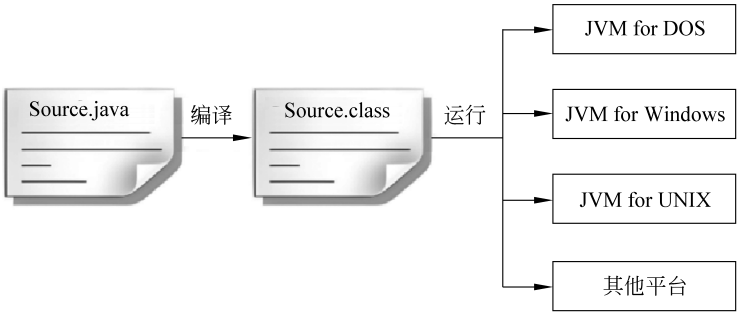

图 1-1　Java 语言的工作机制

2. 运行阶段：JVM 驱动字节码执行

字节码的执行由 Java 虚拟机(JVM)主导。不同平台(如 Windows、Linux、macOS)的 JVM 虽然实现细节不同,但都遵循相同的字节码规范。运行时,JVM 首先加载对应的 .class 文件,并对字节码进行严格的验证,确保其安全性和合法性。之后,JVM 通过两种方式将字节码转换为本地机器指令：对于普通代码,采用解释执行的方式;对于热点代码(频繁执行的部分),则通过即时编译(JIT)技术将其编译为高效的本地机器码。这种混合执行模式既保证了 Java 程序的灵活性,又提升了其执行效率。通过 JVM 这一中间层,Java 程序无须针对不同平台进行修改,即可实现“一次编写,到处运行”的跨平台特性。

Java 的“一次编写,到处运行”理念,就是依托于以上机制实现的。开发者无须针对不同系统分别编写和编译代码,只需一次编译成字节码,各平台 JVM 就能解析并运行。开发者只需关心 Java 语和标准库。不同平台的 JVM 屏蔽了底层差异统一的字节码是实现跨平台的基础。

1.2　搭建 Java 开发环境

1.2.1　下载和安装 JDK

开发和运行 Java 程序,第一步便是下载并安装 JDK(Java development kit,Java 开发工具包)。JDK 包括了编译器(javac)、Java 虚拟机(JVM)以及许多开发 Java 程序所需的核心工具和类库。正确安装 JDK 是学习和使用 Java 语言的基础。

1.2.1　下载和安装 JDK

如图 1-2 所示,在 Oracle 公司的网站可以下载 JDK 的最新版。在下载页面上,根据自己的操作系统(如 Windows、Linux 或 macOS)选择对应的 JDK 版本下载相应安装文件即可。

下载完成后,双击安装文件 exe 文件,出现 JDK 安装向导界面,如图 1-3 所示,按照提示进行安装。大多数情况下,只需保持安装向导的默认选项并不断单击“下一步”按钮即可。进入安装路径设置界面,可以更改 JDK 默认安装目录,如图 1-4 所示。安装完成后单击“关闭”按钮,即可完成 JDK 安装,如图 1-5 所示。

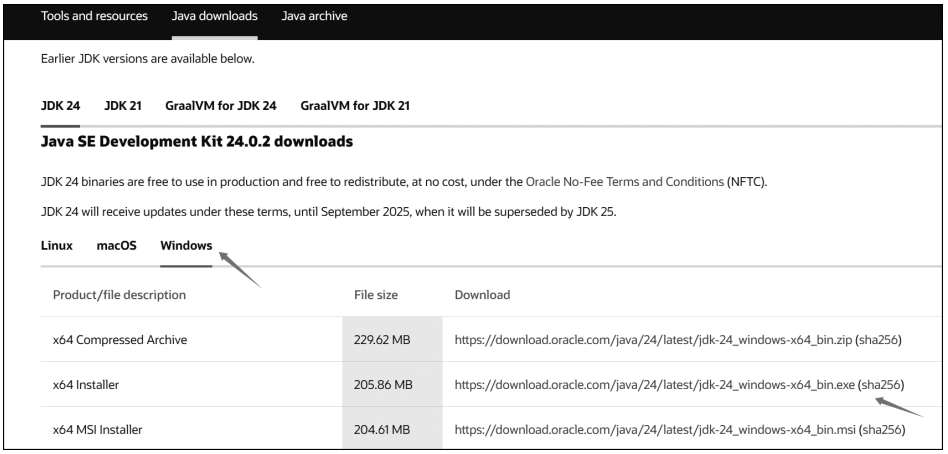

图 1-2　Oracle 官网下载 JDK

图 1-3　JDK 安装向导界面

图 1-4　JDK 安装路径设置界面

图 1-5　JDK 安装成功界面

1.2.2　下载和安装 IntelliJ IDEA

1.2.2　下载和安装 IntelliJ IDEA

IntelliJ IDEA 是由 JetBrains 公司精心打造的一款功能强大的 Java 集成开发环境(IDE)。自诞生以来,IntelliJ IDEA 以卓越的智能代码助手、灵活的自动补全、强大的代码重构能力以及对最新主流开发框架的广泛支持,赢得了全球开发者的喜爱。在实际开发中,IntelliJ IDEA 不仅能够帮助程序员快速准确地编写和维护代码,还通过丰富的插件生态系统,适配企业级、移动端和 Web 应用等多种开发场景。无论是初学者深入学习 Java,还是专业团队进行复杂项目开发,IntelliJ IDEA 都能为用户带来高效、流畅和富有创新力的开发体验,成为现代 Java 开发领域不可或缺的得力助手。

我们可以进入官网主页,单击导航栏的 Download 按钮获取 IDEA 的安装程序。

(1) 进入 JetBrains 官网主页,单击导航栏的 Developer Tools 选项,可以找到 JetBrains 的所有开发工具,选择 IntelliJ IDEA,如图 1-6 所示。

(2) 进入 IntelliJ IDEA 产品页面,单击 Download 按钮,进入 IntelliJ IDEA 下载页面,如图 1-7 所示。

(3) 在下载页面,IDEA 有两个版本,具体如下。

① 社区版(community edition):免费供用户使用,适合大多数个人开发者和开源项目,提供了基础的开发工具。

② 旗舰版(ultimate edition):这是付费版本,支持更多企业级开发和高级特性,比如数据库、Spring 全家桶、企业级框架等。

选好版本后,还需注意选择适合自己操作系统的安装包。通常官网会自动检测你的系统类型,并高亮显示相应的下载按钮。如果没有自动检测,可以手动选择 Windows、Linux 或 macOS。确认无误后,单击 Download 按钮即可开始下载。整个下载过程通常较为简便,速度取决于你的网络环境。

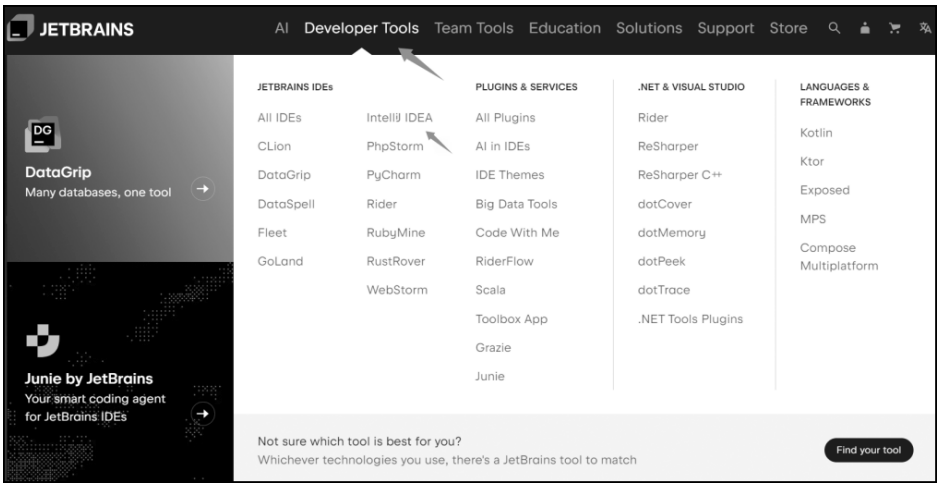

图 1-6　JetBrains 官网主页

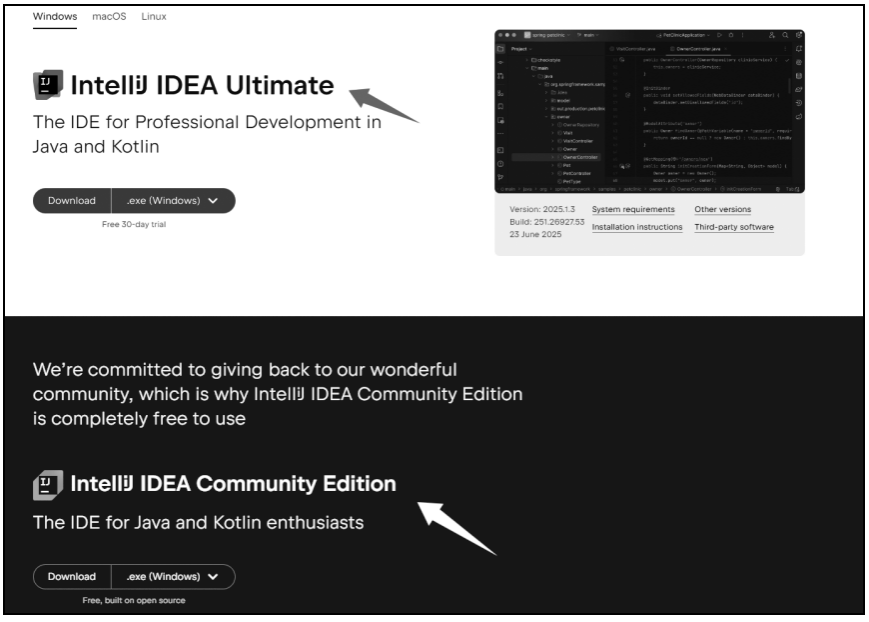

图 1-7　IntelliJ IDEA 下载页面

下载完成后，在本地找到安装包文件 ideal C-2025.2.exe，按照常规安装软件的方式双击运行，跟随安装向导完成所有步骤。至此，就完成了 IntelliJ IDEA 的下载准备，可以正式进入 Java 开发学习和实践环节了。

1.2.3　编写第一个 Java 程序

1.2.3　编写第一个 Java 程序

1. 新建项目

打开 IDEA 窗口，在 IDEA 的菜单栏中选择 File→New→Project 命令，弹出 New Project 对话框。在 Name 文本框和 Location 文本框中输入项目名和项目路径，并在 JDK 下拉列表中选择要使用的 JDK 的版本，最

后单击 Create 按钮完成项目的创建,如图 1-8 所示。

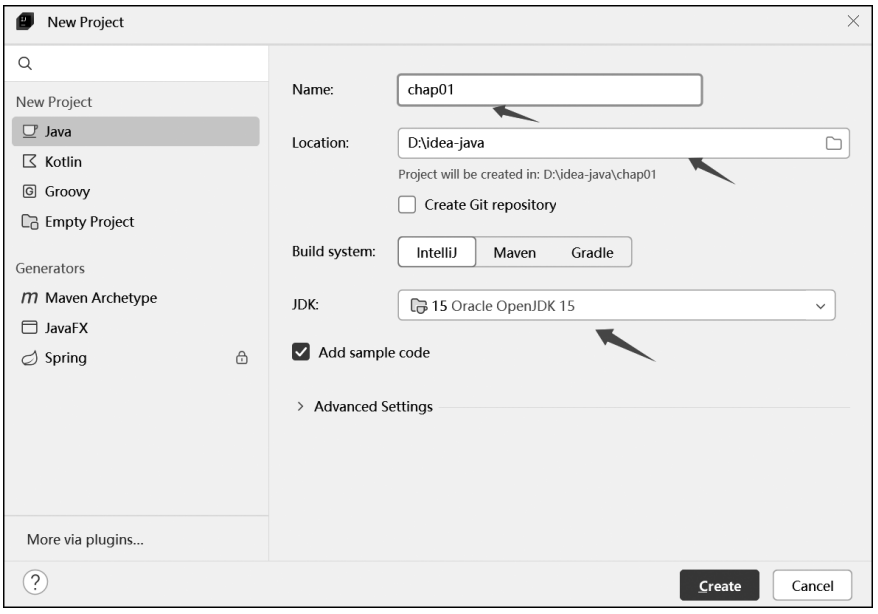

图 1-8　新建 Java 项目

2. 新建类

在项目名称下的 src 目录上右击,在弹出的快捷菜单中选择 New→Java Class(新建类)命令,如图 1-9 所示,在弹出的 New Java Class(新建 Java 类)对话框中输入要新建的类名称(HelloWorld)。

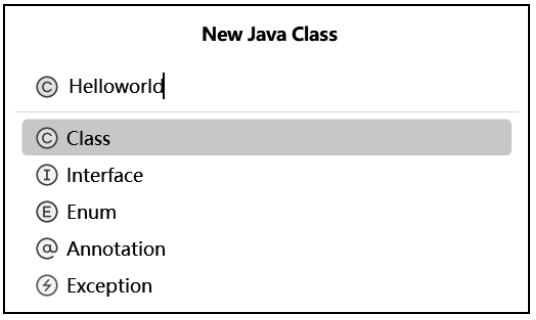

图 1-9　New Java Class 对话框

类名输入完后按 Enter 键,会在程序编辑窗口自动生成类的声明代码。

3. 运行程序

如图 1-10 所示,在代码编辑区输入程序代码,输入完成后,执行 Run 命令运行该程序。也可直接在代码编辑区右击,在弹出的快捷菜单中选择 Run HelloWorld 命令,程序会自动编译并运行,程序运行结果会出现在 Run 窗口中,至此,使用 IDEA 编写并运行完第一个 Java 程序。

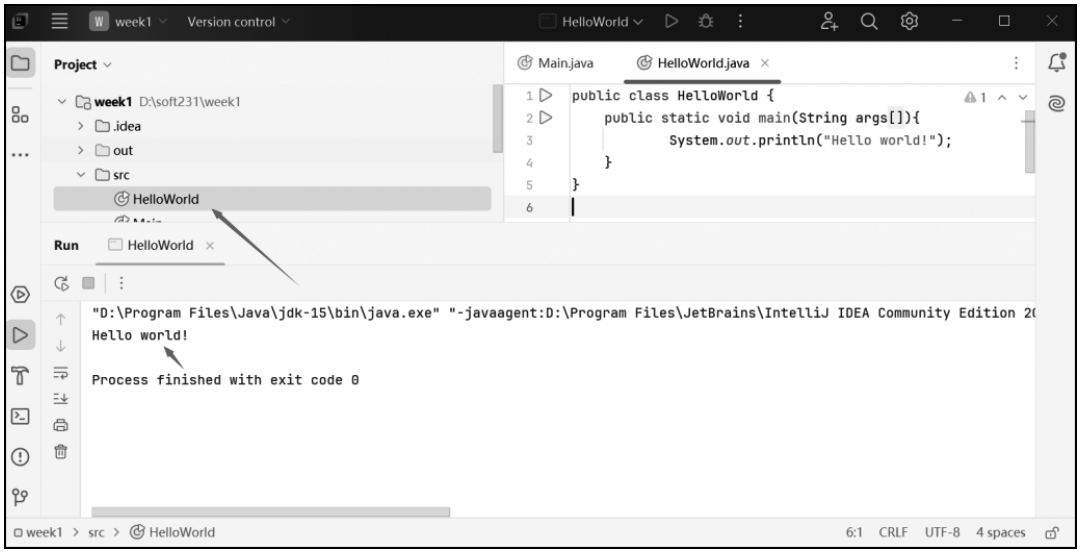

图 1-10　IDEA 程序编辑窗口

1.3　AI 辅助编程

1.3.1　AI 辅助编程简介

AI 辅助编程(AI-assisted programming)是指利用人工智能技术增强软件开发流程,通过机器学习、自然语言处理和大数据分析,为开发者提供智能化的编码支持。这一技术并非为了取代程序员,而是作为“智能协作者”全面提升开发效率和代码质量。

1.3　AI 辅助编程

目前,AI 辅助编程主要以四种形式深度融入开发流程:一是 IDE 智能插件(如 GitHub Copilot 等),在编码过程中实时提供行级补全和文档自动生成;二是自动化代码审查工具(如 Amazon CodeGuru),能够通过静态分析检测代码缺陷并提出性能优化建议;三是低代码/无代码平台(如 Microsoft Power Apps),支持用自然语言需求直接生成可视化逻辑组件,大幅降低开发门槛;四是对话式编程系统(如 ChatGPT),实现从架构设计到调试的全流程自然语言交互,促进需求到代码的智能转化。这些工具共同构建了覆盖编码、测试与维护全生命周期的智能辅助矩阵,使开发者能够更加专注于高价值的创新工作。

近年来,国内也涌现出如阿里云通义灵码、百度 Comate、华为 CodeArts Snap 等本地化 AI 辅助工具,广泛支持主流 IDE,并在代码生成与智能补全等方面表现突出。

本书将以 IDE 智能插件——CodeGeeX 为例,系统介绍 AI 辅助编程的典型应用与实践。

1.3.2 CodeGeeX 的应用

作为国内自主研发的领先代码生成模型之一,CodeGeeX 是由清华大学与智谱 AI 联合研发的先进 AI 辅助编程工具,不仅支持多种主流编程语言(如 Java、Python、C++ 等),还能够无缝集成到 Visual Studio Code、IntelliJ IDEA 等常用 IDE 环境中。它具备出色的上下文理解能力,能够根据开发者当前的代码内容和意图,智能补全函数,生成代码片段,并实现不同语言之间的代码互译。同时,CodeGeeX 还支持对已有代码逻辑进行详细解释,帮助开发者更好地理解和优化程序。值得一提的是,CodeGeeX 基于开源协议发布,提供模型和插件的开放下载,极大促进了社区参与和产品持续迭代。下面让我们进一步看看如何在实际开发环境中应用 CodeGeeX。

1. 安装 CodeGeeX 插件

选择 File→Settings 命令,在打开的 Settings 对话框中选择 Plugins→Marketplace 选项卡,可以看到如图 1-11 所示界面。

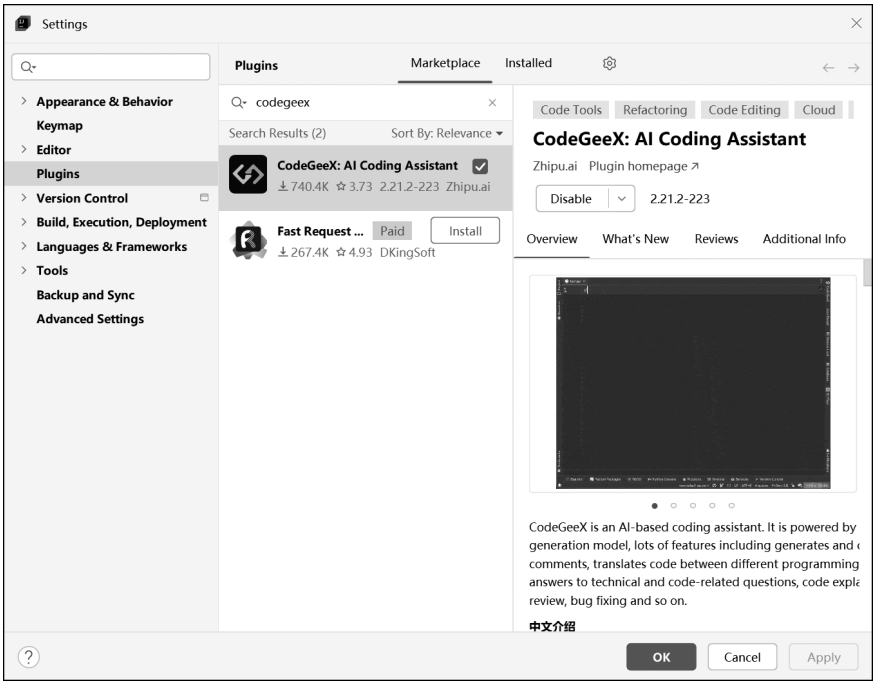

图 1-11 安装 CodeGeeX 插件

单击 Install 按钮,可以安装 CodeGeeX,安装完成后重启 IDEA。

2. CodeGeeX 常用功能

(1) 代码自动补全。CodeGeeX 可以帮助开发者自动补全代码。当编写代码时,它会实时分析当前的上下文,并提供相关的代码补全建议。只需按下 Tab 键即可选择建议。

```
1. public class HelloWorld {
2.   public static void main(String[] args) {
3.     //CodeGeeX 会自动补全这里的代码
```