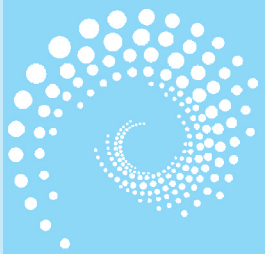
第1章

概　述

CHAPTER 1

本章导读

在线练习



1.1 引言

近年来，在国际权威杂志 *IEEE Spectrum* 发布的关于“Top Programming Languages”（最热门的编程语言）的报告中，Java 语言往往排在第 2 位或第 3 位，与 Python、SQL 和 C++等同时被列为最热门和最有前景的编程语言。详细信息可以查阅杂志官网报道。

在中国，Java 语言的相关生态非常完备，基于 Java 的企业级应用系统非常普遍，每年也都有大量的本科或硕士毕业生应聘 Java 软件工程师等岗位。

Java 的主要优势，不仅是语言本身（特别是语法）简单易学，其类库更是异常丰富，这极大地方便了程序员开发安全和高效的复杂应用系统。同时，众多以 Java 为主要语言的企业级应用系统（例如“双 11”系统）也以稳健著称。

在 AI 时代，因为 AI 模型训练、微调和调用的需要和普及，Python 语言受到了前所未有的关注和欢迎。但 Java 作为面向企业级应用系统开发的编程语言（而非脚本语言），在很多领域，仍然拥有着不可替代的地位。近年来，为了满足在 Java 应用系统中植入 AI 的需求，各种同步/异步、本地/远程等调用 AI 模型的方式也被人们所使用。

AI 时代的来临，对于在大学阶段学习 Java 知识和技能的学生提出了更高的要求，不仅要掌握语言和类库，还要掌握开发“植入 AI 的应用系统”的能力，而这也是本书题目的由来——Java 大学教程：语言、类库与 AI 应用。

1.2 Java 简史

早在 1991 年，在 Patrick Naughton 和 James Gosling 的带领下，太阳计算机系统公司（Sun Microsystems）的工程师开始研发一门新的跨平台和可移植的编程语言。1995 年，在 SunWorld’95 大会上，HotJava 浏览器发布。1996 年，JDK 1.0 正式发布。随后经历了多个版本的更新。在 2009—2010 年间，Sun Microsystems 被 Oracle 收购。2011 年，Java SE 7 发布。2014 年，Java SE 8 发布，这是一个具有里程碑意义的版本，包含了非常大的改进，也是一个长期支持（Long-Term Support，LTS）版本。从 2018 年开始，每年有两个 JDK 版本发布，包括 4 个 LTS 版本，分别是 2018 年的 JDK 11、2021 年的 JDK 17、2023 年的 JDK 21 和 2025 年的 JDK 25。本书就是基于 JDK 25 来编写的。

1.3 Java 的特点

作为一门深受用户喜爱的编程语言，Java 主要有以下特点。

（1）跨平台。将 Java 源代码（.java 文件）编译后得到与平台（操作系统和芯片）无关的字节码（.class 文件），并交由 Java 虚拟机（Java Virtual Machine，JVM）负责从加载到执行的整个生命周期。不同的平台需要不同的 JVM，但是 Java 源文件无须重新编译，即编译后的字节码文件可以在不同平台的 JVM 上运行，这就是所谓的“一次编写，到处运行”（Write Once，Run Anywhere，WORA），也被称为跨平台特性。我们可以看到，Java 的跨平台强调

的是可运行。

(2) **可移植**。在跨平台的基础上,Java 程序在不同平台上运行的结果是一致的。为了实现可移植性,Java 语言采取了很多措施,包括标准化的基本数据类型(Primitive Data Type),例如,int 的大小在所有平台上都是 32 位。我们发现,可移植不仅要求可运行,还要求运行结果一致。

(3) **面向对象**。Java 不是纯面向对象的语言,因为 Java 保留了基本数据类型、类中的静态成员变量和静态成员方法、面向 String 对象的操作符(+)重载等。但是,Java 是以面向对象为核心的语言,与面向对象相关的知识也是在大学阶段学习 Java 语言的重点,例如,类与对象(第 3 章)、继承(第 4 章)、访问权限(第 5 章)、抽象类与接口(第 6 章)、泛型(第 7 章)等。

(4) **简单易学**。Java 的基本语法(第 2 章)与 C++类似,并且 Java 没有指针和 goto 语句,不需要程序员自己管理内存。同时,Java 提供了丰富的类库,例如,字符串(第 8 章)、数据结构(第 9 章)、I/O(第 10 章)、线程(第 11 章)、网络(第 12 章)、数据库(第 13 章)等供程序员调用。

Java 在安全、生态(技术、工具、框架、社区、商业等)、AI 应用(第 14 章和第 15 章)等方面也有诸多特点和优点,鼓励感兴趣的同学查阅相关资料,不断提升自己。

1.4　Java 的开发环境

1.4.1　集成开发环境

我们虽然可以通过文本编辑器编写 Java 程序,并通过 javac 和 java 等命令编译和运行 Java 程序,但是通过集成开发环境(Integrated Development Environment,IDE)可以极大地提升效率。在业界,最主流的 IDE 是 IntelliJ IDEA,它功能强大、生态完善,且相关支持体系完备。在高校,Eclipse 是最为经典和普遍使用的 IDE,也非常适合初学者使用。

关于集成开发环境的安装方法,详见附录 A。

1.4.2　JDK

顾名思义,Java 开发工具包(Java Development Kit,JDK)是开发 Java 程序所必需的工具。

关于 JDK 的下载和安装方法,详见附录 B。

以 JDK 25 为例,在安装好的 JDK 根目录下,我们可以看到如下文件夹。

- ./bin/:包含 javac.exe 和 java.exe 等工具。
- ./conf/:包含一些配置文件。
- ./include/:包含一些 C/C++的头文件(.h 文件)。
- ./jmods/:包含模块化文件(.jmod 文件)。
- ./legal/:包含一些版权和协议方面的法律文档。
- ./lib/:包含 JVM 预加载的类列表(classlist)、JVM 配置文件(jvm.cfg)、JVM 本地

链接库文件(jvm.lib)、源代码(src.zip)等。

1.5 第一个 Java 程序

为了让同学们对 Java 编程有一个直观感受，我们先来看一个简单的源代码示例 A.java：

```
public class A
{
   public static void main(String args[])
   {
      System.out.println("Hello Java!");
   }
}
```

这个示例中有一个类 A，它包含一个主方法(Main Method)，主方法中包含一条语句(Statement)，在 IDE 中运行程序会得到输出“Hello Java!”。也可以在命令行窗口中通过输入 javac A.java 和 java -cp. A 命令得到输出结果。其中“-cp.”表示当前目录为类路径。需要说明的是，以往主方法必须用 public、static 和 void 修饰，是程序的入口。但是在 JDK 25 之后，允许将以上代码简化为如下形式：

```
void main(String args[])
{
   System.out.println("Hello Java!");
}
```

以上简化后的代码无论是在 IDE 中还是通过命令编译和运行，都能得到正确的结果。

当一个源文件中有多个类时，最多只能有一个用 public 修饰的类(也可以没有)，如果有则文件名必须与 public 类相同。

1.6 AI 时代的 Java 编程

在技术突飞猛进和应用生态瞬息万变的 AI 时代，很多事情需要重新审视，特别是知识的学习和技能的掌握，包括 Java 程序设计。为此，在本节，我们将怀着谦卑和谨慎的态度，探讨一些在学习 Java 程序设计时可能会碰到的问题，期望起到抛砖引玉的作用。

在探讨以下问题之前，我们先提出一个基本假设：AI 模型可以通过生成(非复杂)代码等方式帮助程序员大幅提高开发应用系统的效率，但是不能完全取代程序员，因为 AI 模型生成的代码不能保证 100%正确，也不能保证相关设计是最合理、最高效的。不管是普通的信息管理系统，还是安全攸关的控制系统，或是隐私敏感的信息处理系统，都不可能将所有开发、维护和更新等任务完全交由 AI 模型完成。当然，在 AI 模型的辅助下，随着程序员效率的大幅提升，企业所需的程序员数量，特别是初级程序员的数量，无疑会大幅减少。

1.6.1　在 AI 时代，学习 Java 程序设计有何意义？

在 AI 时代，各行各业应用系统的开发、维护和更新依然是一个客观存在且非常普遍的需求，而 Java 语言无疑是已有的和新开发的企业级应用系统中使用占比最高的语言之一，企业必然也有相应的人才招聘和储备需求。因此，从人才供给的角度来看，在高校开设和学习 Java 程序设计方面的课程仍然是非常必要和有意义的。

1.6.2　在 AI 时代，如何开设 Java 程序设计课程？

在 AI 时代，特别是鉴于大语言模型在生成（非复杂）代码等方面的卓越能力，如何在高校开设 Java 程序设计等相关课程是一个不可回避的既紧迫又现实的问题。根据个人的授课经历，在开设相关课程时，需要留意并有意识地强调以下几方面的转变。

（1）从零碎（Fragment）到整体（A Big Picture），例如，从孤岛式、碎片化的 Java 语言知识点到 Java 语言、类库和应用等的全局视图。

（2）从接受“是什么”（What）到分辨“为什么”（Why），例如，从语法规则到语法背后的初衷和动机等。

（3）从记住（Memorize）知识点到学会查阅（Consult）官网文档和相关资源。

（4）从善于回答（Answer）问题到善于向 AI 提问（Ask）。

（5）从个人和团队（Individual ＋ Team）编写代码的能力到 AI 辅助（Individual ＋ Team ＋ AI）编写代码（甚至引导 AI 编写绝大部分代码）的能力。

（6）从侧重编写代码（Write）的能力到兼顾编写代码、调试代码和阅读/审查代码（Write & Debug & Read/Review）的能力，特别是阅读 JDK 源码和审查 AI 生成的可能含有 Bug 的代码。

（7）从编写单一（Simple）功能代码的能力到开发复杂（Complex）应用系统的能力。

（8）从开发应用系统（Application System）的能力到开发（植入 AI 的）智能应用系统（AI-Embedded Application System）的能力。

（9）从单师（Single Teacher）教学到多师（Multiple Teacher）教学，特别是邀请一线企业工程师参与实践教学环节。

（10）在作业和练习中，从考察（Test）学生对知识点和技能的掌握到激发（Spark）学生的兴趣和潜力。

（11）从培养初级程序员（Junior Programmer）到软件工程师（Software Engineer）。

（12）从培养学生学会编程（Learn to Program）到引导他们学会学习（Learn to Learn），并实现不断成长。

1.6.3　在 AI 时代，如何学习 Java 程序设计？

在 AI 时代，学习 Java 程序设计的途径越发多样化，包括：①线下课堂和教材；②Oracle 官网文档；③在线教学视频（例如学堂在线、中国大学 MOOC 等）；④在线评测系统（Online Judgment）；⑤AI 模型（人机互动）；⑥技术社区、博客和微信公众号；⑦GitHub 开源项目；⑧编程类的学科竞赛等。其中，AI 模型不仅可以回答问题和生成代码，还可以出题。同学

们需要根据自身特点选择最适合自己的资源组合，更加高效地掌握相关知识和技能，尽早在大学毕业前跨越初级程序员的阶段。

1.6.4 在 AI 时代，在学习 Java 的过程中碰到问题怎么办？

在 AI 时代，面对学习 Java 程序设计过程中碰到的问题，同学们除了通过传统的方式寻求解决方案外，还应充分利用 AI 模型。需要留意的是，AI 模型给出的答案可能不是最优的（如过于复杂），特别是当提示词（Prompt）不是很合理的时候。因此，同学们需要在与 AI 的互动中学习如何完善提示词，以最终得到最合理的答案或代码。

从与单个 AI 模型互动到与多个 AI 模型互动，是一种非常实用且被广泛使用的方法。例如，将一个 AI 模型生成的代码交由另一个 AI 模型进行校核，往往可以有效降低代码出错的概率。

此外，还需要通过 Oracle 官网文档等来分辨和检验 AI 模型给出的答案有没有包含幻觉、Bug 或过时的内容等。

1.6.5 在 AI 时代，如何让自己成为一个更具竞争力的 Java 程序员？

在 AI 时代，为了增强自身在 Java 程序设计方面的竞争力，同学们可以在大学阶段重点关注以下几方面：①精通 Java 语言基础知识和常用类库；②熟练使用 AI 模型和相关辅助编程工具；③通过在线刷题、开源项目二次开发等方式不断提升代码能力；④通过企业实习或复杂应用系统开发等方式提升工程能力。

1.6.6 在 AI 时代，有了大模型，还需掌握 Java 语言的基础知识吗？

这其实是一个非常容易让初学者产生误解的问题，答案当然是需要的，而且是十分必要的。首先，如果没有扎实的 Java 语言基础知识，就不可能善用 AI 模型，因为连向 AI 提出合理的问题都会变得很困难，自然也无法分辨和判断 AI 生成代码的质量。其次，Java 丰富的类库资源都是建立在各种基础知识之上的，如果没有基础知识作为能力底座，那么在深入学习类库底层实现逻辑时，很容易迷失方向。

1.7 本章小结

本章主要介绍了 Java 语言的一些基本情况，包括发展历史、主要特点、开发环境和简单例子，并重点讨论了在 AI 时代与 Java 编程有关的若干问题。需要说明的是，在 AI 浪潮的冲击下，很难准确判断 Java 软件工程师岗位和职业前景，同学们需要积极拥抱变化、勇于乘势而上。

习题

1. 查阅资料，简述 JRE 与 JDK 的联系与区别。

2. 下载、安装最新的 LTS(Long-Term Support)版本的 Java SE Development Kit，进行 JRE/JDK、系统环境变量等的设置(如需要)，之后进行简单的测试以确认安装成功。

3. 下载、安装 Eclipse IDE for Java Developers(最新版)，并进行 JRE/JDK 的设置(如需要)。

4. 编写一个简单的 Java 应用程序，并在 Eclipse 或 IntelliJ IDEA 集成开发环境中进行编译和运行。

5. 查阅资料，从安全的角度阐述 Java 无指针运算和内存自动管理带来的益处。

6. 查阅资料，简要介绍 Java 的发展史(1000 字以内)。

7. 浏览相关资料，阅读“What's New”或“Security”中的内容，并用自己的话进行介绍(500～800 字)，要求重点突出、条理清楚，可读性强。

8. 查阅两本 Java 经典教材“LIANG Y D, LIAL M L. Introduction to Java Programming and Data Structures: Comprehensive Version[M]. 13th ed. London: Pearson, 2023.”和“HORSTMANN C S, CORNELL G. Core Java, Volume I: Fundamentals[M]. 13th ed. Sebastopol: Oracle Press, 2024. & HORSTMANN C S, CORNELL G. Core Java, Volume Ⅱ: Advanced Features[M]. 13th ed. Sebastopol: Oracle Press, 2024.”的一级目录，分析两本教材的异同点，要求重点突出、条理清楚。

9. 查阅资料，阐述如何在 AI 时代成为一个有竞争力的 Java 软件工程师(300～500 字)。要求逐点简要阐述，重点突出，条理清楚。

10. 收集 Java 程序设计方面的在线学习资源，例如习题、在线编程练习等，详见随书资源。

11. 查阅常见的 AI 辅助编程工具(例如通义灵码、GitHub Copilot、Cursor 等)，分析它们的优缺点，并以其中的 1 个工具为例展示其使用方式和效果。

12. 查阅资料，阐述“Python+Java”的开发范式，即如何使用 Python 语言构建 AI 服务(包括数据处理、模型训练和微调等)，并在开发基于 Java 的企业级应用系统中调用 AI 服务。