Java程 序 设 计

第 1 讲： Java 语言入门与开发环境搭建

厦门大学嘉庚学院

XIAMEN UNIVERSITY TAN KAH KEE COLLEGE

CONTENTS

Java 的诞生与发展历程

语言定位：1995 年 Sun 发布（现 Oracle），具面向对象、跨平台、自动 GC 与原生并发特性的工业级语言。

核心技术特征

- **面向对象**：类与对象纯粹封装机制
- **原生跨平台**：“一次编写，到处运行”（JVM）
- **内存安全**：GC 自动垃圾回收，无指针风险
- **高并发支持**：多线程与 Java 21 虚拟线程
- **庞大生态**：全球第一企业级后端基石

重要 LTS 长期支持版演进

- **Java 8** (2014)：Lambda 表达式、Stream 流
- **Java 11** (2018)：模块化系统、HttpClient
- **Java 17** (2021)：Record 记录类、密封类
- **Java 21** (2023)：**当前主力 LTS**，虚拟线程
- **Java 25** (2025)：新一代 LTS 长期支持版

Java 的工业应用全景格局

企业级中后台服务
Spring Boot / 微服务架构

大数据处理与流计算
Kafka, Flink, Spark

物联网与通信控制中台
Netty, 高并发通信网关

Android 移动开发
移动智能终端系统

金融与核心高可用系统
银行清结算、交易核心

工业控制与仿真软件
智能监控、调度系统

【知识要点】核心认知

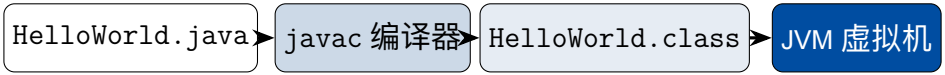
Java 是构建大型、分布式、高并发、长期稳定运行系统的行业首选基石语言。

JDK、JRE 与 JVM 的三层包含关系

【知识要点】核心区别

- **JVM**：负责将字节码翻译为特定操作系统机器码；
- **JDK = JRE + 开发工具**。开发者只需安装 JDK 21 即可兼具开发与运行能力。

跨平台运行原理:“一次编写,到处运行”



C 语言:针对机器编译

源码直接编译为目标 CPU 机器码。不同操作系统系统调用与不同 CPU 架构指令集不同,必须重新交叉编译。

Java 语言:针对 JVM 编译

编译生成的是中立的**字节码(.class)**。由安装在特定操作系统上的 JVM 负责翻译为本机机器码!

随堂互动 1：概念抢答与辨析

【随堂抢答·概念检测】问题：关于 JDK、JRE 与 JVM 的描述，以下哪一项是正确的？

A. 只要运行 Java 编译好的 `.class` 程序，就必须在电脑上安装完整的 JDK。

B. JVM 负责把 `.java` 源代码文件直接编译为机器能看懂的二进制机器指令。

C. Java 能够跨平台是因为不同操作系统上运行着对应平台的 JVM，而字节码文件（`.class`）是跨平台通用的。

D. Windows 平台编译生成的 `.class` 文件不能直接复制到 Linux 机器上运行。

随堂互动 1：答案与深度解析

【知识要点】正确答案：【C 】

逐项解析：

- **A 错误**：仅运行已编译的程序只需 JRE（或包含 JVM 的运行时），开发才必须装 JDK；
- **B 错误**：把 `.java` 编译为 `.class` 字节码的是 `javac` 编译器，而不是 JVM；
- **C 正确**：Java 跨平台的核心是“中立的字节码 + 各平台专属的 JVM 虚拟机”；
- **D 错误**：字节码格式完全标准统一，Windows 下生成的 `.class` 复制到 Linux 上的 JVM 依然可以无缝运行！

课程离线安装包资源说明 (package/)

【知识要点】离线安装包路径说明

为方便在机房离线环境快速配置，本课程项目目录下直接提供了官方离线安装包：

- **Windows 64 位**：`package/jdk-21_windows-x64_bin.zip` (约 196MB)
- **Linux x64**：`package/jdk-21_linux-x64_bin.tar.gz` (约 198MB)
- **macOS 用户**：可使用 Homebrew 安装或下载对应 PKG。

核心环境变量职责

- **JAVA_HOME**：指向 JDK 的解压根目录（如 `C:\Program Files\Java\jdk-21`）；
- **Path**：向系统声明命令搜索路径，加入 `%JAVA_HOME%\bin` 后，即可在任意命令行直接执行 `javac` 与 `java`。

Windows 环境变量图形化配置四步法

【代码演练】Windows 平台实操步骤

1. **解压安装包**：将 `package/jdk-21_windows-x64_bin.zip` 解压至纯英文目录（如 `D:\Java\jdk-21`）；
2. **新建 JAVA_HOME**：【系统变量】新建 `JAVA_HOME`，变量值设为 JDK 解压根目录；
3. **配置 Path 变量**：编辑系统变量 `Path`，新增 `%JAVA_HOME%\bin` 并移至首行；
4. **终端命令验证**：新开终端窗口执行 `java -version` 与 `javac -version` 确认版本输出。

【代码演练】Linux 终端配置命令

```
# 1. 解压安装包到 /opt/java
sudo mkdir -p /opt/java
sudo tar -zxvf package/jdk-21_linux-x64_bin.tar.gz -C /opt/java/

# 2. 配置环境变量（写入 ~/.bashrc 或 /etc/profile）
echo 'export JAVA_HOME=/opt/java/jdk-21' >> ~/.bashrc
echo 'export PATH=$JAVA_HOME/bin:$PATH' >> ~/.bashrc
source ~/.bashrc

# 3. 验证版本输出
java -version
javac -version
```

【代码演练】HelloWorld.java

```java
// 单行注释：这是我们的第一个 Java 源码
/*
 * 多行注释：HelloWorld 示范
 */
public class HelloWorld {
    // 主方法：程序的唯一执行入口
    public static void main(String[] args) {
        // 向标准控制台终端打印一行文本并换行
        System.out.println("Hello, Java World!");
    }
}
```

类名规则

`public class ...`
公开类名必须与文件名完全一致（区分大小写）

main 方法签名

`public static void main...`
固定写法，JVM 识别的程序唯一入口点

标准输出

`System.out.println`
打印内容并自动追加换行符

命令行编译与运行全流程

【代码演练】在终端执行编译与运行指令

```
# 1. 进入 HelloWorld.java 所在目录
cd tex-university/code/lec01/

# 2. 编译：将 .java 编译为 .class 字节码
javac HelloWorld.java

# 3. 观察目录下生成的 HelloWorld.class 文件
ls -l HelloWorld.class

# 4. 运行：启动 JVM 执行字节码（注意：绝不能写成 HelloWorld.class !）
java HelloWorld
```

【知识要点】预期控制台输出

```
Hello, Java World!
```

命令行参数传递机制 (String[] args)

【知识要点】main 方法形参 String[] args 的作用

String[] args 用于在命令行执行程序时向程序内部传递动态配置参数(如用户姓名、学号或调试开关)。

```java
public class ArgsDemo {
    public static void main(String[] args) {
        System.out.println(">>> 接收到命令行参数个数: " + args.length);
        for (int i = 0; i < args.length; i++) {
            System.out.printf("参数 [%d]: %s\n", i, args[i]);
        }
    }
}
```

```
$ javac ArgsDemo.java
$ java ArgsDemo 张三 20240101 --debug
>>> 接收到命令行参数个数: 3
参数 [0]: 张三
参数 [1]: 20240101
参数 [2]: --debug
```

【随堂找茬·避坑挑错】看看以下代码和执行指令,分别隐藏了哪些典型错误?

```java
// 代码片段 1:
public class test { // 文件名为 Test.java
    public static void Main(String args) {
        System.out.println("Hello") ; // 注意这里的括号和分号
    }
}
```

执行指令 2 报错现象

```
$ java HelloWorld.class
```

错误: 找不到或无法加载主类 HelloWorld.class

【知识要点】错误点全景剖析

1. **类名大小写不匹配**：`public class test` 与 `Test.java` 大小写不一致，编译直接报错；
2. **main 方法签名错误**：`Main` 首字母必须小写，且参数必须是数组 `String[] args`；
3. **全角中文符号**：）是全角中文括号，Java 语法要求全半角英文字符；
4. **运行指令误带.class**：`java` 命令执行的是**类名**，不是文件名，正确写法是 `java HelloWorld`！

交互式编程利器：jshell REPL 工具

【知识要点】什么是 jshell?

JDK 9+ 提供的原生交互式编程环境（REPL）。无需创建类与 main 方法即可直接执行 Java 代码片段，是轻量验证语法与 API 的利器。

```
$ jshell
|  欢迎使用 JShell (JDK 21) -- 输入 /help 获取帮助
jshell> int a = 10, b = 20;
a ==> 10
b ==> 20
jshell> a + b
$3 ==> 30
jshell> System.out.println("Hello JShell!");
Hello JShell!
jshell> /exit
|  再见
```

三种注释格式

- **单行注释**：// 注释内容
- **多行注释**：/* 多行内容 */
- **文档注释**：/** 文档注释 */
  （由 javadoc 提取生成 API 文档）

常用 Javadoc 标签

- @author：代码作者
- @version：版本号
- @param：形参含义说明
- @return：返回值说明
- @throws：异常抛出说明

```java
/**
 * 学生实体信息类
 * @author 张振
 * @version 1.0
 */
public class Student {
    /**
     * 计算学生加权平均绩点
     * @param scores 各门功课成绩
     * @return 加权平均分
     */
    public double calcGpa(double[] scores) {
        return 0.0;
    }
}
```

随堂实操练笔 1：名片卡生成器 (CommWelcome.java)

【随堂实操·动手练笔】实操任务 1：编写个人名片生成程序

编写 CommWelcome.java：从命令行接收姓名与学号两个参数（args[0]，args[1]）；参数不足时给出用法提示并退出；在控制台格式化打印个人名片卡。

```java
public class CommWelcome {
    public static void main(String[] args) {
        if (args.length < 2) {
            System.out.println("用法: java CommWelcome <姓名> <学号>");
            return;
        }
        System.out.println("========= 个人信息名片卡 =========");
        System.out.printf("  * 姓    名 : %s\n", args[0]);
        System.out.printf("  * 学    号 : %s\n", args[1]);
        System.out.println("  * 课程名称 : Java程序设计（通信工程）");
        System.out.println("================================");
    }
}
```

随堂实操练笔 2：jshell 交互式算术验证

【随堂实操·动手练笔】实操任务 2：使用 jshell 快速验证表达式

请在各自电脑终端打开 jshell，依次输入并观察输出：

1. 计算表达式：`100 + 200 * 3`
2. 定义字符串：`String course = "Java 程序设计";`
3. 调用字符串方法：`course.length()` 与 `course.toUpperCase()`
4. 退出 jshell：输入 `/exit`

工程视野与自主可控

【工程视野】自主可控与开放标准

Java “一次编写，到处运行”彰显开放标准价值；认识底层虚拟机构筑自主可控软件生态的核心战略意义。

【实战技巧】代码审查与辅助开发

实战建议：遇终端报错时，可将报错堆栈发给 AI 并提示：“请解释这行报错的底层成因并提供修复范例。”

【随堂实操与课后思考】课后思考与巩固

- **练习**：独立在个人电脑完成 JDK 21 环境变量配置、命令行编译与 jshell 交互练习。
- **思考**：C 语言直接编译为机器码，Java 编译为字节码在 JVM 上运行，两者在性能与跨平台灵活性上如何权衡？

课程互动与讲师主页

在线主页与课程资源

- **个人主页**：
  `https://matnoble.top`
- **课程专栏**：
  `https://matnoble.top/courses/`
- **GitHub 仓库**：
  `https://github.com/MatNoble`

思考：Questions & Answers

“学贵善疑，逻辑筑基”

感谢大家的专注聆听！

课件最新版本与补充资料将持续同步至课程主页。

现在进入自由提问与交流时间。