

Java程 序 设 计

第 1 讲： Java 语言入门与开发环境搭建

- 1 本讲目标
- 2 循环一:为什么学 Java
- 3 循环二:从源码到运行
- 4 循环三:第一个程序
- 5 循环四:报错不慌
- 6 机动:效率工具与速查
- 7 收束:自查与课后

本讲目标:下课前,你应该能做到

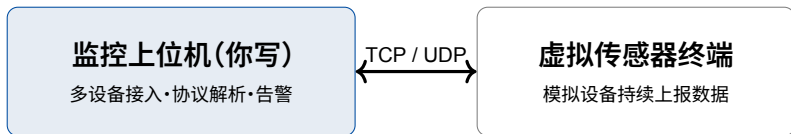
三件事

1. **会跑**:用命令行把 HelloWorld.java 编译并运行起来;
2. **会讲**:说清 .java → .class → JVM 这条链,Java 凭什么跨平台;
3. **会查**:独立排查初学者最常见的 4 类报错。

课堂节奏

- 4 个循环,每个循环:讲 10 分钟 → 动手 5 分钟;
- 抢答先自己猜,再验证;实操限时完成,截图发课程群。

先看终局:16 周后你交出的东西



- 第 24 讲验收:你的程序同时连多台虚拟设备,解析协议帧、超限告警、断线重连;
- 到达路径:语法(1-8 讲)→ 面向对象(9-14 讲)→ 网络通信(15-21 讲)→ 集成答辩(22-24 讲)。

Java 是什么,通信工程为什么用它

语言定位

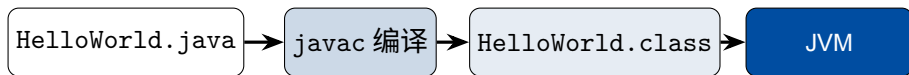
- 1995 年诞生,全球企业级后端第一语言;
- 面向对象、跨平台、自动内存回收;
- 本课程用 **JDK 21** (长期支持版)。

一句话:要建**长期稳定运行的大型系统**,Java 是行业首选——这正是通信系统对软件的要求。

你身边用它的地方

- 电信级通信网关与物联网中台 (Netty);
- 大数据:Kafka、Flink、Spark 全是 Java 系;
- 银行核心、Android、工业监控软件。

跨平台:一次编写,到处运行



Windows / Linux / macOS 各有专属 JVM

- C 语言:直接编译成**本机机器码**,换平台必须重新编译;
- Java:编译成中立的**字节码**,由各平台的 JVM 翻译执行——**字节码通用,JVM 不通用**。

JDK、JRE、JVM：三层包含关系



- JVM: 把字节码翻译成操作系统的机器码;
- 开发者装一套 JDK = 编译 + 运行全都有。

【随堂抢答·概念检测】以下哪一项是正确的？

- A. 只要运行编译好的 .class 程序,就必须安装完整 JDK
- B. JVM 负责把 .java 源代码直接编译成机器指令
- C. 跨平台靠的是:字节码通用 + 各平台专属的 JVM
- D. Windows 编译的 .class 不能复制到 Linux 上运行

先独立写下答案——30 秒后验证。

【知识要点】逐项排查

- A 错:只运行程序,JRE 就够;开发才需要 JDK;
- B 错:编译 .java 的是 javac,JVM 只负责执行字节码;
- C 对:中立字节码 + 平台专属 JVM,跨平台的全部秘密;
- D 错:字节码标准统一,复制到任何平台都能跑。

```
public class HelloWorld {           // 类名 = 文件名
    public static void main(String[] args) { // 入口
        System.out.println("Hello, Java World!");
    }
}
```

- **类名规则**: public 类名必须与文件名完全一致(区分大小写);
- **main 签名**: public static void main(String[] args) 固定写法, JVM 的唯一入口;
- **输出**: println = 打印 + 换行。

命令行:编译与运行是两条命令

```
$ cd code/lec01/  
$ javac HelloWorld.java      # 编译: 生成 HelloWorld.class  
$ ls HelloWorld.class  
$ java HelloWorld           # 运行: 写类名, 不带 .class  
Hello, Java World!
```

- javac 吃的是**文件名**(带 .java);
- java 吃的是**类名**(不带 .class)——写错立刻见报错。

实操 1:跑通你的第一个程序

【随堂实操·动手练笔】HelloWorld ·限时 3 分钟

打开终端,进入 `code/lec01/` 目录:

1. `javac HelloWorld.java` 编译;
2. `java HelloWorld` 运行。

完成标志:控制台输出 `Hello, Java World!` → 截图发课程群,举手。

main 的参数:程序从外部收数据

```
public class ArgsDemo {  
    public static void main(String[] args) {  
        System.out.println(args.length);  
        for (int i = 0; i < args.length; i++)  
        {  
            System.out.println(args[i]);  
        }  
    }  
}
```

运行效果

```
$ java ArgsDemo 张三 20240101  
个数: 2  
0: 张三  
1: 20240101
```

String[] args 接收启动命令后面的所有词。

实操 2: 给程序递一张名片

【随堂实操·动手练笔】ArgsDemo ·限时 2 分钟

编译并运行：

1. `javac ArgsDemo.java;`
2. `java ArgsDemo 你的名字你的学号。`

完成标志:控制台逐行打印出你传的两个参数 → 截图发课程群。

找茬 1:这段代码加这条命令,几处错?

【随堂找茬·避坑挑错】文件存为 Test.java,运行报“找不到主类”——找找看

```
public class test {  
    public static void Main(String args) {  
        System.out.println("Hello") ;  
    }  
}
```

```
$ java Test.class
```

```
错误: 找不到或无法加载主类 Test.class
```

30 秒,能找出几处?

【知识要点】初学者四大坑

1. 类名 `test` 与文件名 `Test` 不一致(大小写敏感);
2. `Main` 应为 `main`,且参数必须是 `String[] args` 数组;
3. **全角括号**):中文标点是编译错误的头号来源;
4. `java` 后面写类名,不能带 `.class`。

实操 3: 亲手制造两个报错

【随堂实操·动手练笔】报错脱敏·限时 3 分钟

在 `code/lec01/` 里故意制造并观察报错：

1. 把 `HelloWorld` 类名改成小写再编译——读一遍报错；
2. 执行 `java HelloWorld.class`——读一遍报错；
3. 改回去，重新编译运行确认恢复。

完成标志：亲眼见到两种报错并恢复原状 → 举手。

jshell:不用写类,直接算

```
$ jshell
jshell> 100 + 200 * 3
$1 ==> 700
jshell> "Java".length()
$2 ==> 4
jshell> /exit
```

- JDK 自带的交互式环境:敲一行出一行,验证语法最快捷;
- 注意:100 + 200 * 3 先乘后加——运算符优先级在哪儿都生效。

三种注释

- 单行:// 注释;
- 多行:/* 多行 */;
- 文档:/** */——javadoc 命令能提取成 API 手册。

```
/**  
 * 计算加权平均绩点  
 * @param scores 各门功课成绩  
 * @return 加权平均分  
 */  
public double calcGpa(double[] scores) {  
    return 0.0;  
}
```

@param / @return 写给调用的人看——本课程所有方法的注释标准。

环境速查卡(带回家照着配)

```
# 离线安装包 (课程目录 package/ 下)
#   Windows: jdk-21_windows-x64_bin.zip  解压到纯英文目录
#   Linux:   jdk-21_linux-x64_bin.tar.gz  解压到 /opt/java

# Windows 两个环境变量:
#   JAVA_HOME = JDK 解压根目录
#   Path      追加 %JAVA_HOME%\bin

# 验证 (新开终端):
$ java -version    # 输出 java version "21.x.x" 即成功
$ javac -version
```

机房已配好;个人电脑照此卡配置,5 分钟搞定。

1. **会跑**: HelloWorld 你亲手编译运行成功了吗?
2. **会讲**: .java → .class → JVM, 一句话说清跨平台?
3. **会查**: 类名、main 签名、全角符号、.class 后缀——四个坑都见过实物了?

没做到的那条, 就是课后任务的第一优先级。

工程视野

字节码 + JVM 的架构是“开放标准”的胜利: 自己掌握运行时底座, 才有自主可控的软件生态。

1. **练习**:在个人电脑上照速查卡配好 JDK 21,把 ArgsDemo 跑通;
2. **AI 实战**:把今天遇到的任意一条报错原文发给 AI,让它解释成因并给修复范例,核对它说得对不对。

下讲预告:数据类型——为什么 $0.1 + 0.2$ 不等于 0.3 ?

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>

自由交流与提问 (Q&A)

“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。