

## #Java环境配置

1. JDK是Java的开发工具包，负责将“与平台无关的Java字节码”（.class文件）翻译成当前操作系统能识别的机器码，最终让程序在电脑上执行。

JRE是Java运行的环境，运行已编译的Java程序，JRE = JVM + 核心类库（Java API） + 其他运行支持文件，其中核心类库如 `java.lang`、`java.io`、`java.util` 等包），提供字符串处理、IO操作、集合管理等基础功能

JVM能把Java的字节码转成机器码再交给CPU执行的虚拟机，使之可以在不同平台上运行；

管理程序运行时的内存（如堆、栈、方法区）、垃圾回收（自动释放无用内存）、线程调度等底层资源；

屏蔽不同操作系统（Windows/macOS/Linux）的底层差异。

JDK包含JRE，JRE包含JVM，

一个Java程序从“编写”到“运行”，恰好需要三者协同：

### 1. 开发阶段（依赖JDK）：

开发者用文本编辑器写Java源码 → 用JDK中的javac编译器把源码编译成.class字节码文件；

### 2. 运行阶段（依赖JRE + JVM）：

使用者双击程序或通过命令启动 → JRE中的java命令启动JVM → JVM加载.class字节码并翻译成机器码 → 调用JRE中的核心类库完成功能（如网络请求、数据处理） → 程序运行。

总的来说是JDK负责“造程序”，JRE负责“跑程序”，JVM负责“解程序”

```
C:\Users\ZZF>java -version
java version "24.0.2" 2025-07-15
Java(TM) SE Runtime Environment (build 24.0.2+12-54)
Java HotSpot(TM) 64-Bit Server VM (build 24.0.2+12-54, mixed mode, sharing)

C:\Users\ZZF>javac -version
javac 24.0.2
```

### 2. 环境变量配置了JAVA-HOME和Path中的%JAVA-HOME%\bin

因为当我们在命令行里输入一个命令（像 `java`、`javac` 这些）时，系统得知道去哪里找能执行这个命令的程序文件。环境变量里的 `Path` 就像是一张“地图”，它里面存着一些文件夹的路径，系统会按照这些路径的顺序，去逐个文件夹里找我们输入的命令对应的可执行文件。

把这个 `bin` 文件夹的路径添加到 `Path` 环境变量里后，当在命令行输入 `java` 或者 `javac` 时，系统就会去这个 `bin` 文件夹里找到对应的程序，然后运行它，这样就能在命令行使用这些命令了。


3.

```
C:\Users\ZZF>javac C:\Users\ZZF\IdeaProjects\Java02\src\com\Example\HelloJava.java

C:\Users\ZZF>java C:\Users\ZZF\IdeaProjects\Java02\src\com\Example\HelloJava.java
Hello World

C:\Users\ZZF>
```

 HelloJava

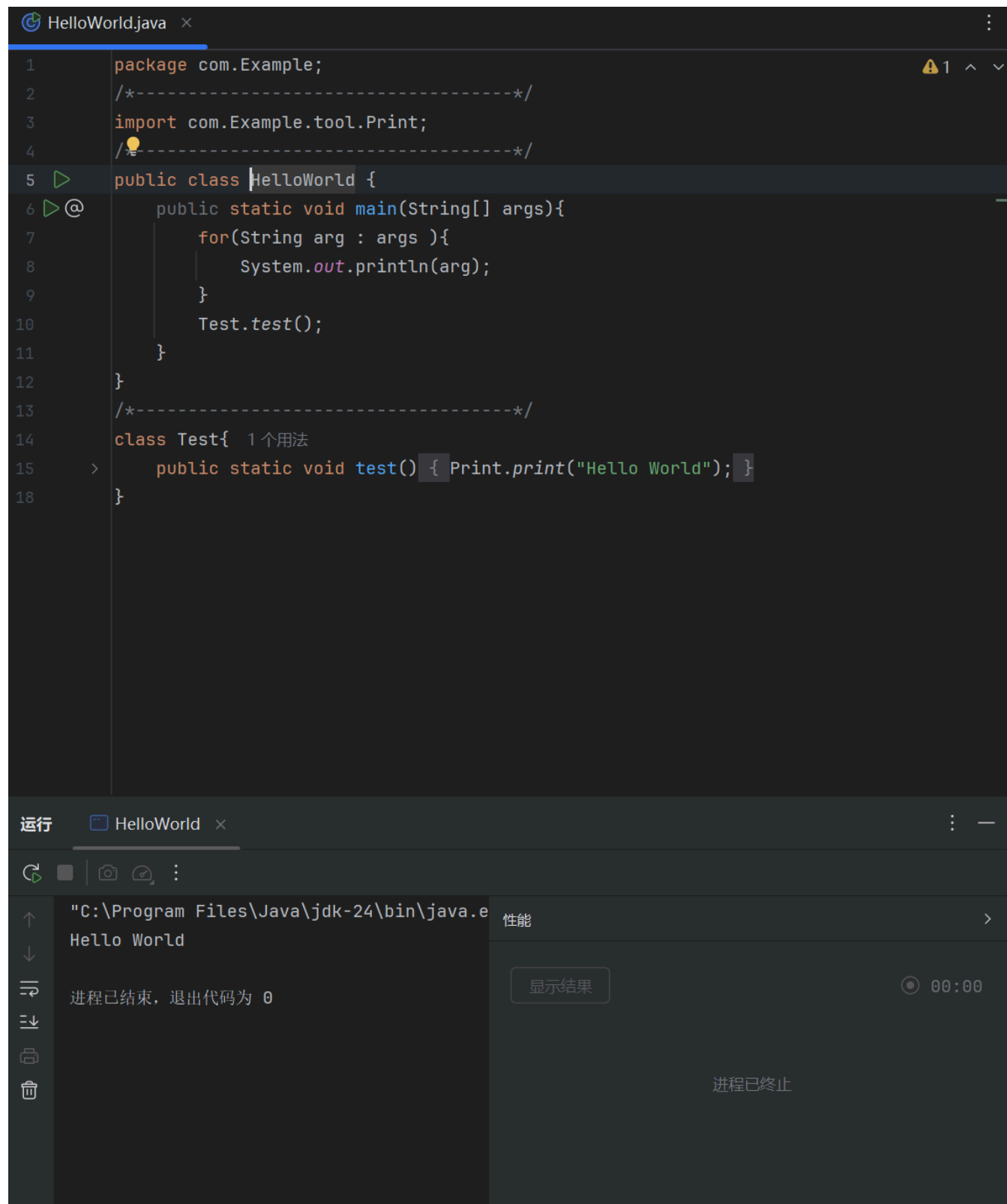
 HelloJava.class

#### 4.文件有源文件.java文件

是 Java 程序的原始代码形式，开发者通过编写类、方法、变量等结构来实现具体功能。编译器会读取源文件中的代码，按照 Java 语法规则进行检查和处理。

#### 字节码文件.class文件

JVM 的类加载器会将字节码文件加载到内存中，经过解释器或即时编译器（JIT）处理后，转化为机器码，交由计算机的 CPU 执行。



The screenshot displays an IDE interface with two main sections. The top section shows the source code of a Java file named `HelloWorld.java`. The code defines a package `com.Example`, imports `com.Example.tool.Print`, and contains two classes: `HelloWorld` and `Test`. The `HelloWorld` class has a `main` method that iterates over command-line arguments, prints them, and calls `Test.test()`. The `Test` class has a `test` method that prints "Hello World".

```
1 package com.Example;
2 /*-----*/
3 import com.Example.tool.Print;
4 /*-----*/
5 public class HelloWorld {
6     public static void main(String[] args){
7         for(String arg : args ){
8             System.out.println(arg);
9         }
10        Test.test();
11    }
12 }
13 /*-----*/
14 class Test{ 1个用法
15     public static void test() { Print.print("Hello World"); }
16 }
17
18 }
```

The bottom section shows the execution output. The command executed is `"C:\Program Files\Java\jdk-24\bin\java.e`. The output is `Hello World`. The process has ended with an exit code of 0.

运行 HelloWorld x

↑ ↓ ↺ ↻ 性能 >

"C:\Program Files\Java\jdk-24\bin\java.e  
Hello World

进程已结束，退出代码为 0

显示结果 00:00

进程已终止