Exercise walk-through

Language Translators ▪ 5.2

eXercise

You must be able to

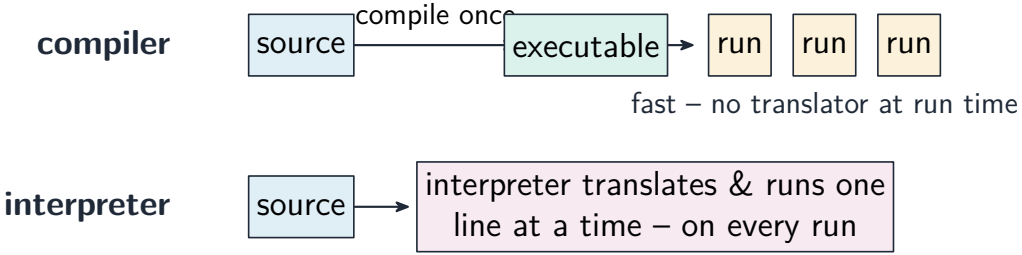
- explain the need for an **assembler**, a **compiler** 编译器 and an **interpreter** 解释器
- compare and justify the use of a compiler or an interpreter
- explain the **hybrid** (bytecode) approach used by Java
- describe features of an **Integrated Development Environment (IDE)** 集成开发环境

Method — Three translators

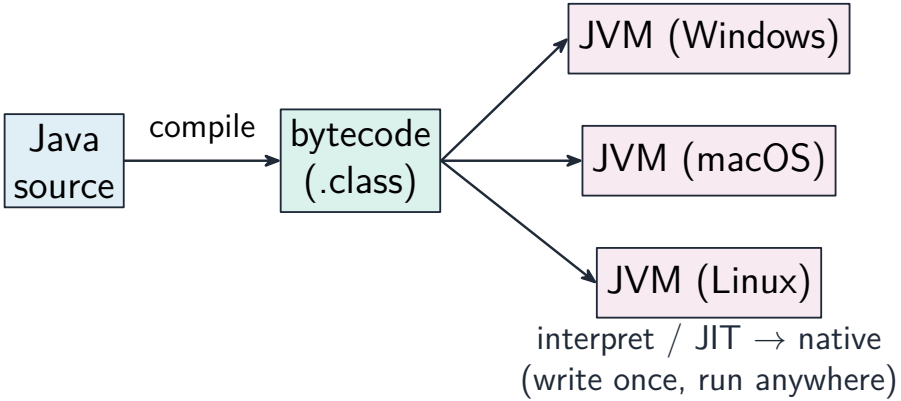
You write source code; the CPU runs **machine code 机器码**. A translator converts between them.

- **assembler** — translates **assembly language** to machine code (one-to-one).
- **compiler** — translates a whole **high-level** program to machine code **once, before it runs**.
- **interpreter** — translates and runs a high-level program **one line at a time**.

Method — Three translators



Method — Three translators



Java compiles to bytecode run by a virtual machine

Method — Compiler vs interpreter

	compiler	**interpreter**
When translated	all at once, before running	line by line, while running
Errors	reports **all** at compile time	stops at the **first** error reached
Output	a stand-alone **executable**	none — re-translates each run
Needs the translator to run?	no	yes
Run-time speed	faster	slower
Portable across platforms?	recompile per platform	yes (if interpreter exists)

Method — Compiler vs interpreter

Choosing: use a **compiler** for speed and to distribute finished software; use an **interpreter** for quick development and cross-platform scripts.

Method — Java (hybrid) and IDEs

Java is **compiled to bytecode 字节码**, which a **virtual machine** (the JVM) interprets (or just-in-time compiles). So errors are caught early **and** the bytecode runs anywhere with a JVM ("write once, run anywhere"). An **IDE** brings the tools together, with features that help at each stage:

- **writing code:** an editor with **syntax highlighting**, **auto-complete** / context-sensitive prompts, and **prettyprinting** (auto-indenting).
- **finding errors: error diagnostics** that point to the line of a syntax error, and reports as you type.
- **debugging:** a **debugger** with **breakpoints**, **single-stepping**, and a **watch window** to inspect variable values while paused.

Practice 2.1 ▪ 2 marks

State the difference between a **compiler** and an **interpreter**.

Solution 2.1

Method: Three translators

Worked steps

A compiler translates the **whole** program **once** into an executable; an interpreter translates and runs it **one line at a time** B1 B1.

Practice 2.2 ▪ 1 mark

Give one situation in which an **interpreter** is more suitable than a compiler.

Solution 2.2

Worked steps

e.g. during development (quick edit–run cycles) / for a cross-platform script / a small or run-once program B1.

Practice 2.3 ▪ 1 mark

State what an **assembler** translates.

Solution 2.3

Method: Three translators

Worked steps

Assembly language (into machine code) B1.

Practice 2.4 ▪ 2 marks

Name **two** features of an IDE.

Solution 2.4

Worked steps

Any two of: syntax highlighting; auto-complete; one-key compile/run; debugger; version-control integration B1 B1.

Exam-style 3.1 ▪ 2 marks

(a) State **two** benefits of using a **compiler** rather than an interpreter.
(b) State **two** benefits of using an **interpreter** rather than a compiler.

Solution 3.1

Worked steps

(a) Any two: runs faster (no translation at run time); produces a stand-alone executable; the translator is not needed to run it; the source code is not given away B1 B1.

(b) Any two: reports errors as it reaches them (easier development); no need to recompile after each change; the same program runs on any platform with the interpreter B1 B1.

Exam-style 3.2 ▪ 3 marks

Explain how Java's "compile to bytecode, then run on a virtual machine" approach lets the same program run on different computers.

Solution 3.2

Method: Three translators

Worked steps

The program is compiled once into **bytecode**, which is platform-independent B1; any computer with a **JVM** can interpret/run that bytecode B1; so it runs unchanged on different operating systems — "write once, run anywhere" B1.

Exam-style 3.3 ▪ 2 marks

Describe **two** features an IDE provides to help a programmer **debug** a program.

Solution 3.3

Method: Java (hybrid) and IDEs

Worked steps

Any two: set **breakpoints** to pause execution; **step through** line by line; **inspect variable** values while paused B1 B1.

Exam-style 3.4 ▪ 2 marks

A company sells finished software to customers who do not have any programming tools installed. State which translator the company should use, and justify your choice.

Solution 3.4

Method: Compiler vs interpreter

Worked steps

A **compiler** B1 — it produces a stand-alone executable that runs without any programming tools installed on the customer's computer B1.

Exam-style 3.5 ▪ 4 marks

A programmer writes a program in a high-level language and releases it as **free software**.
(a) **Explain** how the programmer can use an **interpreter** first and then a **compiler** while developing the same program.
(b) **State** two things an **Integrated Development Environment** provides beyond a translator.
(c) A user downloads the program. **Describe** what "free software" allows the user to do that **freeware** does not.
(d) **Explain** why the program still runs more slowly when interpreted than when compiled, even on the same computer.
(e) The compiler reports a **syntax error** on a line that looks correct. **Suggest** one reason this can happen.

Solution 3.5

Method: Three translators

Worked steps

(a) During development an **interpreter** translates and runs one statement at a time, so the programmer sees the effect of a change immediately and can stop at the first error B1 B1. Once the program works, a **compiler** translates the whole source into machine code once B1, producing an executable that runs faster and can be distributed without the source B1.
(b) Any two: an editor with syntax highlighting and auto-complete; a debugger with breakpoints, single-stepping and variable watches; a project manager for multiple source files; built-in help and error reporting B1 B1.

Solution 3.5

(c) Free software gives the user the source code and the right to **study**, **modify** and **redistribute** it, including modified versions B1 B1. Freeware costs nothing to use but the source is not provided and the licence forbids changing or redistributing it B1.
(d) An interpreter must **translate each statement every time it is executed**, so a statement inside a loop is translated on every pass B1 B1. Compiled code was translated once, before running, so the processor executes machine code directly with no translation overhead at all B1.
(e) The real error is on an **earlier** line — for example a missing closing bracket or quotation mark — and the compiler only notices when the statement fails to make sense B1.