Exercise walk-through

5.2.1 Partial compilation (Java) and the IDE

A-Level Computer Science

You must be able to

- explain how Java compiles to **bytecode 字节码** that a **virtual machine 虚拟机** runs, and why
- describe the IDE features the syllabus names, in its four groups: coding, initial error detection, presentation and debugging
- identify the group a feature belongs to, and describe how it helps the programmer
- describe how the debugging features are used to find an error

1 The method

Method — Partially compiled, partially interpreted

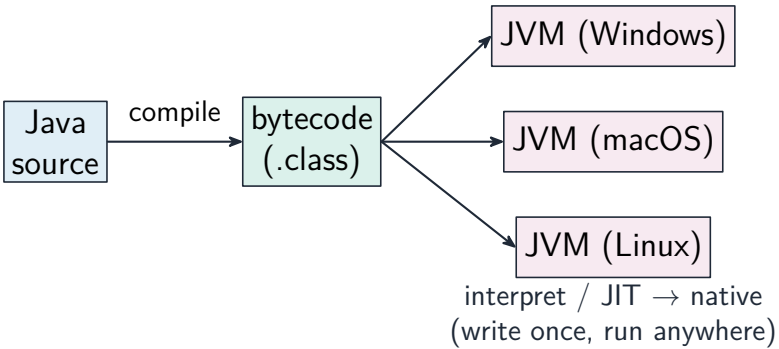
Java is the syllabus's example. The program is translated in **two stages**:

1. A **compiler** translates the source code once into **bytecode**, an intermediate form that is not the machine code of any real processor.
2. The bytecode is run by a **virtual machine** (the JVM), which **interprets** it into the machine code of the computer it is running on.

Why both? A compiler alone produces machine code for **one** processor and operating system; an interpreter alone is slow. Compiling to bytecode does the heavy translation once and reports the syntax errors early, and the **same** bytecode file then runs on any computer that has a JVM: "write once, run anywhere". The price is speed, which a **just-in-time 即时编译** compiler in the JVM recovers by compiling frequently run bytecode into native code while the program runs.

Method — Partially compiled, partially interpreted

Compiling to bytecode does the heavy translation once and reports the syntax errors early, and the same...



Method — The IDE

An IDE is **one application** that brings together the tools for writing, translating, running and debugging a program, so the programmer never leaves it. The syllabus sorts its features into **four groups**.

Method — The IDE

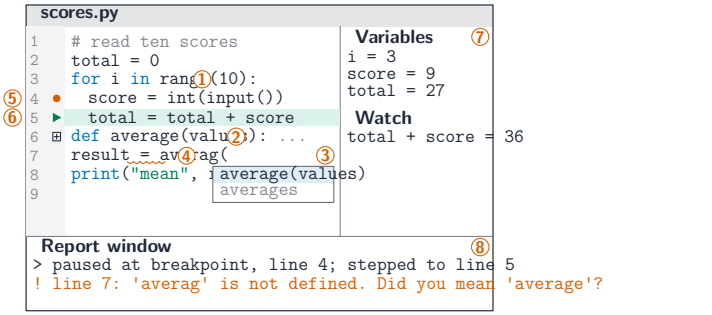
group	feature	what it does for the programmer
coding	**context-sensitive prompts 上下文相关提示**	pops up the identifiers, keywords or parameters that fit at this point; **auto-complete** finishes the name
initial error detection	**dynamic syntax checks 动态语法检查**	checks the syntax **as you type** and underlines a mistake before the program is translated
presentation	**prettyprint 代码美化**	keywords, identifiers and comments in different colours or fonts, with consistent indentation
presentation	**expand and collapse code blocks**	hides the body of a loop, an IF or a subroutine, so the outline is visible
debugging	**breakpoints 断点**	the running program pauses when it reaches a marked line
debugging	**single stepping 单步执行**	from the pause, runs one statement at a time
debugging	a window for **variables and expressions**	shows the current value of each variable, and of any expression you ask for, while paused
debugging	the **report window 报告窗口**	lists errors, warnings and output

Method — The IDE

A question that gives one feature and asks for "one other" from the same group wants a feature from the **same row group**: prettyprint pairs with expand and collapse; single stepping pairs with breakpoints, the variables window or the report window.

Method — The IDE

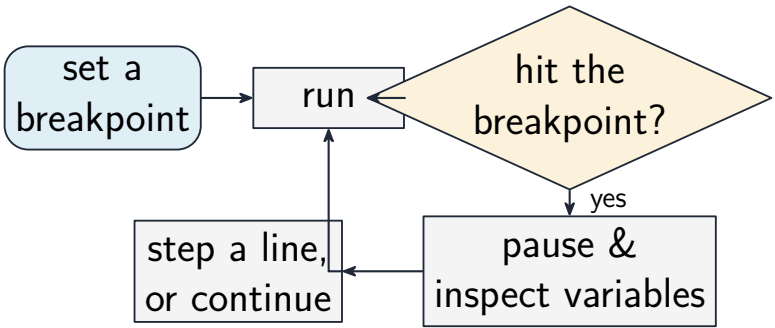
An IDE is one application that brings together the tools for writing, translating, running and debugging a...



Method — Finding an error with the debugger, for four marks

1. Set a **breakpoint** on the first line of the function, so the program pauses there instead of running through.
2. **Single-step** through the function one statement at a time.
3. After each step, read the values in the **variables and expressions** window and compare them with the values you expected.
4. The first statement after which a value is wrong is where the error is; the **report window** shows any run-time error message and the output so far.

Method — Finding an error with the debugger, for four marks



2 Practice

Practice 2.1 ▪ 3 marks

Complete the sentences by writing the missing words.
A Java program is first compiled into ..... . This is then run by a ..... , which ..... it one instruction at a time on the computer it runs on.

Solution 2.1

Method: Partially compiled, partially interpreted — p.4

Worked steps

- bytecode B1; virtual machine (or JVM) B1; interprets B1

Practice 2.2 ▪ 4 marks

Write the letter of the group that each IDE feature belongs to.

letter	**group**
A	coding
B	initial error detection
C	presentation
D	debugging

Practice 2.2 ▪ 4 marks

feature	**letter**
breakpoint	_____
context-sensitive prompt	_____
dynamic syntax check	_____
expand and collapse code blocks	_____
prettyprint	_____
report window	_____
single stepping	_____
variables and expressions window	_____

Solution 2.2

Method: The IDE — p.6

Worked steps

- breakpoint **D**; context-sensitive prompt **A** B1
- dynamic syntax check **B**; expand and collapse code blocks **C** B1
- prettyprint **C**; report window **D** B1
- single stepping **D**; variables and expressions window **D** B1

Practice 2.3 ▪ 3 marks

Name the IDE feature described in each sentence.

Practice 2.3 (a)

As the programmer types `Total.`, a list of the things that can follow appears.

Solution 2.3 (a)

Method: The IDE — p.6

Worked steps

context-sensitive prompt (auto-complete) B1

Practice 2.3 (b)

A wavy red line appears under a line with a missing bracket before the program is run.

Solution 2.3 (b)

Worked steps

dynamic syntax check B1

Practice 2.3 (c)

The programmer clicks a small arrow and the 30 lines of a subroutine are hidden, leaving only its header.

Solution 2.3 (c)

Worked steps

expand and collapse code blocks B1

Practice 2.4 ▪ 1 mark

State what happens when a running program reaches a **breakpoint**.

Solution 2.4

Worked steps

- the program pauses at that line, so the programmer can inspect the variables (and then step or continue) B1

Practice 2.5 ▪ 2 marks

Describe **prettyprint**.

Solution 2.5

Worked steps

- the code is displayed with keywords, identifiers, strings and comments in different colours or fonts B1
- and with consistent indentation, so the structure of the program is visible at a glance B1

Practice 2.6 ▪ 2 marks

State **two** benefits of compiling a program to bytecode rather than to machine code.

Solution 2.6

Method: Partially compiled, partially interpreted — p.4

Worked steps

any **two** of:

- the same bytecode runs on any computer that has a virtual machine B1
- the program does not have to be compiled again for each type of processor or operating system B1
- syntax errors are still found by the compiler before the program runs B1

3 Exam-style

Exam-style 3.1 ▪ 7 marks

A company writes an app in Java. The app must run on phones, laptops and smart TVs, which use different processors and operating systems.

Exam-style 3.1 (a) ▪ 3 marks

Describe how the app is translated and run on each device.

Solution 3.1 (a)

Method: Partially compiled, partially interpreted — p.4

Worked steps

the Java source code is compiled once into bytecode B1

- the bytecode is the same file for every device; each device has its own virtual machine (JVM) B1
- the virtual machine interprets the bytecode into the machine code of that device as the app runs B1

Exam-style 3.1 (b) ▪ 4 marks

Explain one benefit and **one** drawback of this approach compared with compiling the app directly to machine code.

Solution 3.1 (b)

Worked steps

benefit: one compiled file runs on every device, so the app does not have to be compiled separately for each processor and operating system B1 B1

- drawback: interpreting bytecode is slower than running machine code directly, so the app runs more slowly (unless the JVM uses just-in-time compilation) B1 B1

Exam-style 3.2 ▪ 8 marks

For each IDE feature, **state** the group the syllabus places it in and **describe** how it helps the programmer.

Exam-style 3.2 (a)

context-sensitive prompts

Solution 3.2 (a)

Method: The IDE — p.6

Worked steps

coding B1; as the programmer types, the IDE suggests the identifiers, keywords or parameters that fit at that point, so names are typed correctly and quickly B1

Exam-style 3.2 (b)

expand and collapse code blocks

Solution 3.2 (b)

Worked steps

presentation B1; the body of a loop, an IF or a subroutine can be hidden, leaving its header, so the programmer sees the outline of a long program and can find a section quickly B1

Exam-style 3.2 (c)

report window

Solution 3.2 (c)

Worked steps

debugging B1; it lists the errors, warnings and the output of the program in one place, so the programmer sees what went wrong and where B1

Exam-style 3.2 (d)

dynamic syntax checks

Solution 3.2 (d)

Worked steps

initial error detection B1; the syntax is checked as the code is typed and a mistake is underlined at once, so it is corrected before the program is translated B1

Exam-style 3.3 ▪ 4 marks

A function `Average()` should return the mean of a list of marks but returns a value that is too large.
Describe how the programmer uses the debugging features of an IDE to find the line that contains the error.

Solution 3.3

Method: Finding an error with the debugger, for four marks — p.10

Worked steps

- set a **breakpoint** at the start of `Average()`, so the program pauses when the function is called B1
- **single-step** through the function one statement at a time B1
- after each step, read the values of the variables (the total, the count) in the variables window and compare them with the expected values B1
- the first statement after which a value is wrong, for example the total being added to twice or the count being one too small, is the line with the error B1

Exam-style 3.4 ▪ 4 marks

A programmer has to read a program of 2000 lines written by somebody else.
Identify two presentation features of an IDE and **describe** how each one helps the programmer read the program.

Solution 3.4

Worked steps

two features, each described:

- **prettyprint**: keywords, identifiers and comments are shown in different colours, with consistent indentation, so the programmer can see the structure of each part at a glance B1 B1
- **expand and collapse code blocks**: the bodies of subroutines and loops can be hidden, so the programmer reads the outline of the 2000 lines first and opens one part at a time B1 B1

Exam-style 3.5 ▪ 4 marks

A student says: "An IDE is just a text editor with a compile button."
Explain two ways in which an IDE does more than this.

Solution 3.5

Method: Partially compiled, partially interpreted — p.4

Worked steps

any **two**, each explained:

- it detects errors as the code is typed (dynamic syntax checks) and offers context-sensitive prompts, so many mistakes are avoided before compiling B1 B1
- it contains a debugger with breakpoints, single stepping and a variables window, so a logic error can be found by watching the program run B1 B1
- it presents the code with prettyprint and collapsible blocks, and reports errors and output in a report window, so the whole edit, run and fix cycle happens in one place B1 B1