

5.2.1 Partial compilation (Java) and the IDE

Name: _____ Class: _____ Date: _____

Total: 42 marks

KEY WORDS bytecode • virtual machine
 • integrated development environment • breakpoints • single stepping

Objective

Explain Java bytecode and JVM execution; classify IDE features and use a debugger to find an error.

1 Key knowledge and worked examples

Study these first. Each one shows a **method** that a question later on this sheet needs.

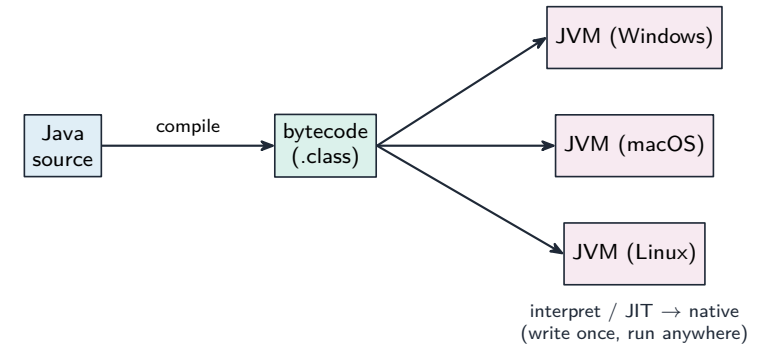
■ Key knowledge: use these earlier methods

- **Licences:** proprietary software restricts copying or modification under its licence. Open-source licences permit source inspection and specified modification/redistribution. Free Software means the freedoms to run, study, modify and share, not merely a zero price; freeware may be free of charge without these freedoms.
- A licence sets the permitted uses. Copyright protects the creator's work; obey both when redistributing a modified program.
- **RLE:** replace a consecutive run by its value and count. It is lossless, but short/no runs can increase file size because counts add overhead.

■ Partially compiled, partially interpreted

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

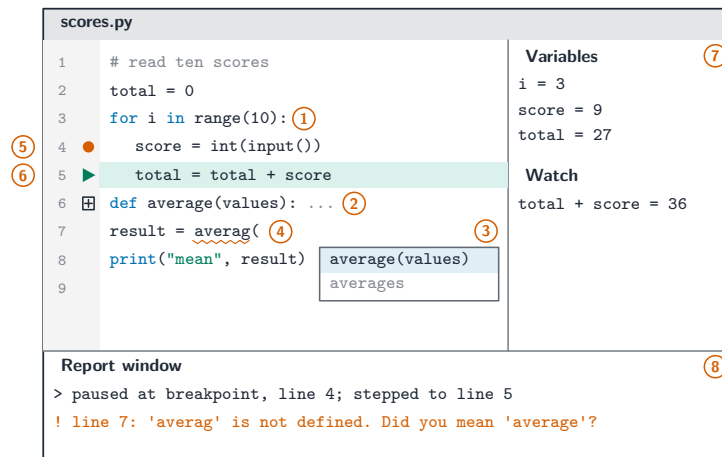
1. Compile source code once into **bytecode**, an intermediate form.
2. A **JVM** interprets bytecode on the target computer; JIT compilation converts frequently run bytecode to native machine code.



Bytecode is portable to devices with a JVM; syntax errors are detected during compilation. JVM execution has overhead compared with native machine code; JIT compilation can improve speed.

■ The IDE

An **IDE** integrates writing, translating, running and debugging tools.

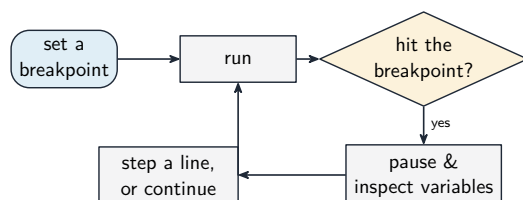


- ① prettyprint: coloured keywords/strings/comments
- ② expand / collapse a code block
- ③ context-sensitive prompt
- ④ dynamic syntax check, flagged as you type
- ⑤ breakpoint: the program pauses here
- ⑥ single stepping: run one line at a time
- ⑦ variables and expressions, watched while paused
- ⑧ report window: errors, warnings and output

group	feature	what it does for the programmer
coding	context-sensitive prompts	suggests valid names or parameters; auto-complete finishes a name
initial error detection	dynamic syntax checks	checks syntax as you type; flags errors before translation
presentation	prettyprint	colours/fonts distinguish code; consistent indentation
presentation	expand and collapse code blocks	hides a block body to show its outline
debugging	breakpoints	the running program pauses when it reaches a marked line
debugging	single stepping	from the pause, runs one statement at a time
debugging	a window for variables and expressions	shows current values while paused
debugging	the report window	lists errors, warnings and output

For “one other feature”, choose from the same group.

■ Finding an error with the debugger, for four marks



A function returns an unexpected value. Describe how the debugging features of an IDE help find the cause.

1. Set a **breakpoint** at the function entry; run until it pauses.
2. **Single-step** one statement at a time.
3. Compare the variables/expressions window with expected values.
4. Identify the first incorrect value; check the report window for errors and output.

2 Practice

Now use the methods above. The first questions need one idea each.

2.1 Complete the sentences by writing the missing words. [3]



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.

2.2 Write the letter of the group that each IDE feature belongs to. [4]



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

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

2.3 Name the IDE feature described in each sentence. [3]

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

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

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

(a) `Total.` (c) `PROCEDURE Show()`

(b) `IF (A > B`

2.4 State what happens when a running program reaches a **breakpoint**. [1]

●

2.5 Describe **prettyprint**. [2]



2.6 State **two** benefits of compiling a program to bytecode rather than to machine code. [2]

bytecode

machine code

3 Exam-style questions

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

(a) **Describe** how the app is translated and run on each device. [3]



Java app phone laptop smart TV

(b) **Explain one** benefit and **one** drawback of this approach compared with compiling the app directly to machine code. [4]

3.2 For each IDE feature, **state** the group the syllabus places it in and **describe** how it helps the programmer. [8]

(a) context-sensitive prompts



(b) expand and collapse code blocks

(c) report window

(d) dynamic syntax checks

3.3 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. [4]



3.4 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. [4]

2000 lines

3.5 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. [4]

[illegible]