

5.2.1 Partial compilation (Java) and the IDE

Name: _____ Class: _____ Date: _____

Total: 42 marks

KEY WORDS bytecode 字节码 • virtual machine 虚拟机

• integrated development environment 集成开发环境 • breakpoints 断点 • single stepping 单步执行

Objective

Learn to answer exam questions on a program that is **partially compiled and partially interpreted**, such as Java, and on the features of an **Integrated Development Environment** 集成开发环境 (IDE). Sheet 5.2 covered compilers and interpreters; this is the last sheet of topic 5.

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

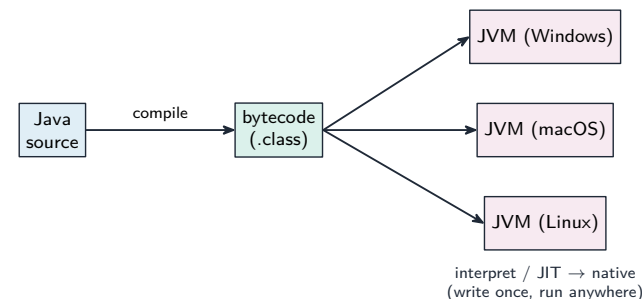
1 Worked examples

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

■ Partially compiled, partially interpreted

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

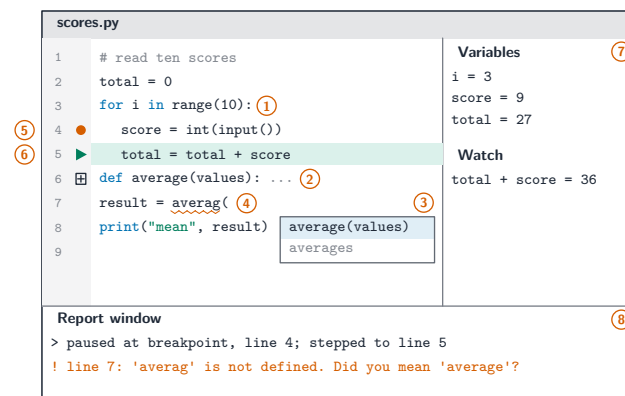
- A **compiler** translates the source code once into **bytecode**, an intermediate form that is not the machine code of any real processor.
- The bytecode is run by a **virtual machine** (the JVM), which **interprets** it into the machine code of the computer it is running on.



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.

■ The IDE

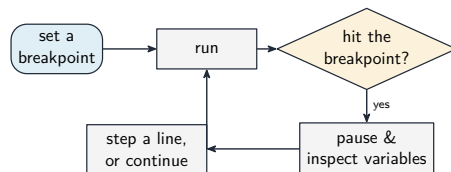
An IDE is **one application** that brings together the tools for writing, translating, running and debugging a program, so the programmer never leaves it. The syllabus sorts its features into **four groups**.



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

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.

■ 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** 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.

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.

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]

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]

(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]

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]

Solutions

Each answer shows the steps, then the **result**.

2.1

- bytecode; virtual machine (or JVM); interprets

2.2

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

2.3

- (a) context-sensitive prompt (auto-complete)
- (b) dynamic syntax check
- (c) expand and collapse code blocks

2.4

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

2.5

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

2.6 any two of:

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

3.1

- (a) the Java source code is compiled once into bytecode
 - the bytecode is the same file for every device; each device has its own virtual machine (JVM)
 - the virtual machine interprets the bytecode into the machine code of that device as the app runs
- (b) benefit: one compiled file runs on every device, so the app does not have to be compiled separately for each processor and operating system
 - drawback: interpreting bytecode is slower than running machine code directly, so the app runs more slowly (unless the JVM uses just-in-time compilation)

3.2

- (a) coding; as the programmer types, the IDE suggests the identifiers, keywords or parameters that fit at that point, so names are typed correctly and quickly
- (b) presentation; 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
- (c) debugging; it lists the errors, warnings and the output of the program in one place, so the programmer sees what went wrong and where
- (d) initial error detection; 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

3.3

- set a **breakpoint** at the start of `Average()`, so the program pauses when the function is called
- **single-step** through the function one statement at a time
- after each step, read the values of the variables (the total, the count) in the variables window and compare them with the expected values
- 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

3.4 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
- **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

3.5 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
- it contains a debugger with breakpoints, single stepping and a variables window, so a logic error can be found by watching the program run
- 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