5 System Software – Part 2

Name: _____ Class: _____ Date: _____

Vocabulary

Write each word three times.

English	**Write it 3 times** (English)		
compiler			
executable			
interpreter			
bytecode			
virtual machine			
integrated development environment			
debugger			
breakpoints			

Recall

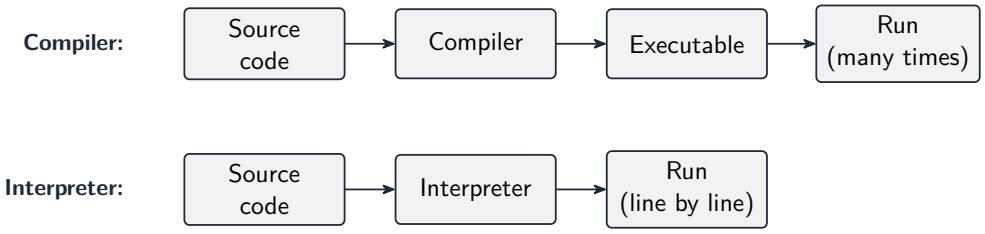
In **Part 1** you met the operating system, utility programs, and the three translators (assembler, compiler, interpreter). Part 2 compares compilers and interpreters more closely, meets the clever "bytecode + virtual machine" idea, and looks at the IDE programmers work in.

New ideas

Read these first. Each idea has a short worked example. Then do the Practice.

1 Compiler vs interpreter

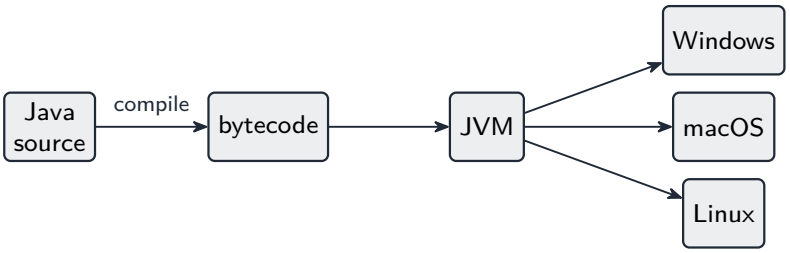
- A **compiler** translates the **whole** program once into a standalone **executable**. It then runs fast and needs no translator present – but it is tied to one type of computer, and it reports all the errors together at the end.
- An **interpreter** translates and runs **one line at a time**. It is slower and must stay installed, but it is easy to edit-and-rerun and to move between computers, and it stops at the first faulty line.



Worked example. A game written in C++ is compiled once and shipped as one fast program; a small Python script is interpreted, so you can change a line and rerun it at once.

2 Bytecode and the virtual machine

Java is compiled into **bytecode** – a platform-independent in-between form. A **virtual machine** (the JVM) on any computer then runs that bytecode. This is "write once, run anywhere": one compiled file runs on Windows, macOS or Linux.

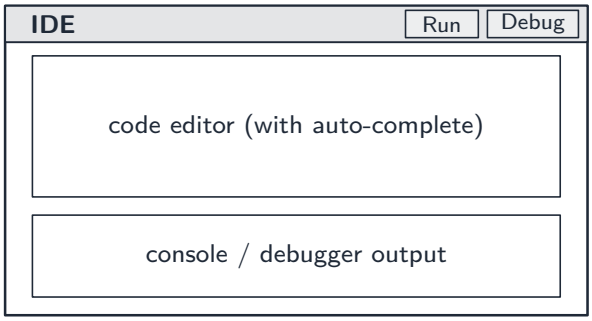


Worked example. A Java app compiled to bytecode on a Windows laptop runs unchanged on a Mac or a phone, because each has its own virtual machine to run that same bytecode.

Watch out: bytecode is *not* machine code – it still needs a virtual machine to run it. That extra step is exactly what lets it run anywhere.

3 The IDE

An **integrated development environment** (IDE) puts all the tools in one program: a code editor with colour **syntax highlighting** and auto-complete, one-key compile-and-run, and a **debugger**. With the debugger you set **breakpoints** to pause the program and inspect the variables, or step through it one line at a time.



Worked example. A breakpoint set on line 12 pauses the program there; you see that `total` is 0 when you expected 50, which points you straight to the bug.

Practice

5.1 State whether a compiled program needs the compiler installed in order to run. [1]

5.2 State what a breakpoint does in a debugger. [1]

5.3 State one difference between a compiler and an interpreter. [2]

5.4 Name two features an IDE provides to help write code. [2]

5.5 Explain why a Java program compiled to bytecode can run on many different computers. [2]

5.6 Explain one reason a debugger with breakpoints can find a bug faster than adding print statements all over the code. [2]

5.7 Challenge. A company wants one app to run on Windows, Mac and Linux without rewriting it. Explain how compiling to bytecode and using a virtual machine makes this possible. [2]