Introduction: Programming and Java

Learning Outcomes:

- ✓ Describe what programming is
- ✓ Identify the different programming levels
- ✓ Identify the purpose of a compiler
- ✓ Describe what Java is
- ✓ Describe the purpose of the JVM

Prerequisite Knowledge:

- ✓ Be able to use a computer i.e. mouse, keyboard and basic operations
- ✓ Have a basic comprehension of maths operations i.e. multiplication, division, addition and subtraction

Keywords

Compiler Bytecode Source Code JVM

0.1 The Theory: Programming and Compilers

What is Programming?

Writing software (computer programs) is a lot like writing down the steps it takes to complete a process. For example, the written steps to making a 'brew' might be simplified to: fill kettle, place kettle on stove, turn stove on and so forth. These written steps, often referred to as instructions, are then passed to a computer where each command is processed in sequence.

High-Level Programming Languages

Most computer programming today is achieved using high-level programming languages. There are lots of these languages available on the market and some are quite old, including COBOL which was devised in the 1950s! More recent and popular languages include Java, Visual Basic, Python, C#, JavaScript and PHP.

A high-level language provides an easy platform to write advanced programs without any concerns to computer architecture, such as specific CPU instructions. These languages come with pre-written, reusable common code known as libraries; programming libraries help to significantly reduce development time. An example of a high-level language is shown below:

Java Code:

```
if (x >= 5) {
      System.out.print("Hello World");
}
```

Explanation:

```
If variable x is greater than or equal
to 5, then print the line 'Hello
World' to the command-line.
```

Programming Tips!

Remember that a computer is dumb, but obedient. In other words, a computer will do exactly what a programmer tells it to do… even if it is not necessarily what the programmer intended it to do. Below are three points to consider when programming:

1. 'Computers do not make mistakes' – programmers do
2. Programming will highlight the importance of 'clarity of expression'
3. Programming instructions are processed in sequence and one at a time

Assembly Languages

An assembly language is one step above a computer's native language, binary (machine language). In an assembly language, instructions are given human-friendly, symbolic names. Unlike high-level languages, a programmer works with basic operations/instructions that a CPU can directly perform, such as comparative operations (i.e. AND, OR, NOT).

Remember, assembly languages are both basic and time-consuming; therefore it is impractical when writing large programs (programs that are normally written using high-level languages). Below is an example of assembly language (notice that the code is not as interpretable as the high-level example):

Programming Tips!

Tips to remember about high-level programming languages (including JAVA) are that:

1. The syntax is sort of like English
2. They have pre-written code called libraries
3. They 'sit on top' of an operating system
4. The syntax is not a language that a CPU understands

Code:	**Explanation:**
`MOV EAX, [EBX]`	Move the 4 bytes stored in memory at the address contained in EBX, into EAX
`MOV [ESI+EAX], CL`	Move the contents of CL into the byte at address ESI+EAX

Machine Language

Computers only understand 'bits' – 0s and 1s, often referred to as binary. Binary is machine language. A computer system uses these bits to represent information, whether that is numbers, characters and/or pixels. The computer also uses bits to represent computer instructions; this is very difficult to achieve, although the pioneers of computer science once did this! However, today most programs are written in high-level languages that are later compiled into machine code (0s and 1s), using a compiler.

Compilers

As stated earlier, computers do not understand programs written in high-level languages, such as C# and Java (they only understand binary, 0s and 1s). High-level languages must be compiled using a compiler to convert the written source code into machine code. Every high-level programming language has a specific compiler; a compiler is a small utility program that is usually packaged with the SDK (Software Development Kit) of each programming language.

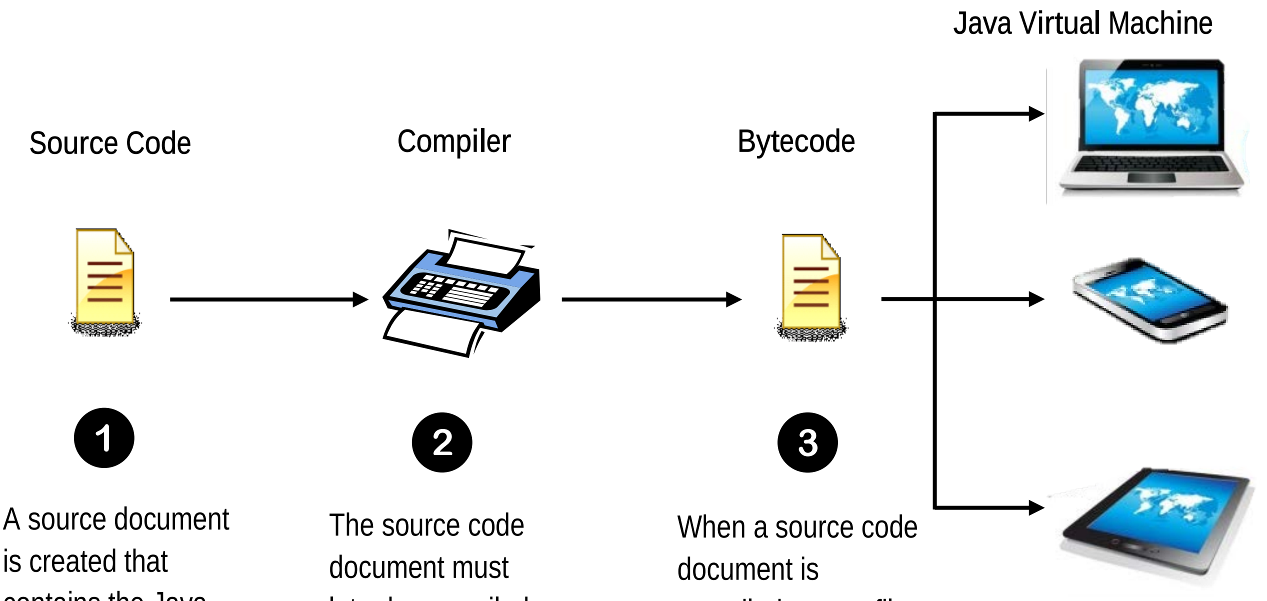
0.2 The Theory: Java

What is Java?

Java is a popular high-level programming language, and platform, that was first released in 1995 by Sun Microsystems. However, Java has since been taken over by the giant Oracle Corporation and has continued to gain popularity; in fact, more than 3 billion devices worldwide run Java. The Java language can be used to create games, database applications (including web-based) and many other enterprising solutions.

How does it work?

Java applications are designed to operate on any platform that has a Java Virtual Machine (JVM) installed; think of the JVM as a software platform to run Java programs on. In theory, a programmer can write a single application in Java, and it will execute similarly on any piece of hardware with an installed JVM, whether it is a mobile device, laptop or tablet. This is achieved by converting source code into bytecode (instructions for the JVM) via the compiler. The JVM on each device can then determine the best way in which to execute the bytecode instructions.



1. A source document is created that contains the Java language – in other words, the application code. The source document is of type *.java; for example, myProgram.java. There are many environments available to create Java files; although the most popular are NetBeans and Eclipse, Java can also be written in Notepad.

2. The source code document must later be compiled using the Java compiler (the javac utility program). The compiler checks (and highlights) the source code document for errors; the document will not be compiled until the compiler is entirely satisfied with the documents' validity.

3. When a source code document is compiled, a new file is created; a Java bytecode file. Java bytecode are instructions that enable any device (with a JVM) to interpret them regardless of their platform. The new bytecode file is of type *.class and will have the same name as the source code file; for example, myProgram.class.

4. The Java Virtual Machine (JVM) is a software platform that enables the Java bytecode (the *.class file) to be executed on most electronic gadgets. This is achieved by converting the bytecode into something meaningful for the underlying platforms.

Activity 8.3

Answer the following questions:

1. What is programming?

   ....

   ....

   ....

   ....

2. What is a high-level programming language?

   ....

   ....

   ....

   ....

3. What is assembly language?

   ....

   ....

   ....

   ....

4. What is machine language?

   ....

   ....

   ....

   ....

5. What is Java? How does it work?

   ....

   ....

   ....

   ....