Head First Java - Chapter 1 Notes

1. How Java Works

- First thing we will need to know is about the way Java works:

The first step is the source code, which is the normal Java code we write.

Then the second step is to enter this code into the compiler, and the result will be bytecode, which is a new document but in .class format.

This .class file will run in any Java Virtual Machine (JVM) to implement the source code, okay?

And Java is different from C++, for example, because in Java, this .class file can literally work in any device and different OS, just the condition is that this device is having a JVM, okay?

That's because the main goal for Java is to "write once and run anywhere", rather than C++ that deals with the device hardware itself.

For example, if you compile the C++ code in Windows, you will get a .exe file.

If you take this and try to run it in Linux, the code will not work, because C++ is dealing with the device internals itself, rather than Java that deals with the abstract JVM layer.

And the JVM is handling its own special device, and that's why we have different and multiple Java Virtual Machines-because each device is different from the others.

There is one that wants optimization, and another wants less storage, and so on.

So we can say the JVM goal is to translates the byte code into something the underlying platform understands and runs your program.

2. Java Speed and Memory

- Okay second thing is what about the speed and memory usage of Java?

In the beginning of Java it was slow. Now, it is not the fastest, but it has become faster, and the magic super power here is the JVM.

The JVM is the abstract layer in Java which deals with the device internals without worrying about it when you write the source code.

Inside this layer, it optimizes the code while it is running. That's why you can create very fast available applications without needing to write special high-performance code like in C++ using MPI.

As we said, Java is more abstracted than C++, but compared to C, Java uses a lot of memory because:

1. When you run Java code, you also run the entire JVM environment, which needs more memory.

2. Java uses an automatic garbage collector that manages memory and checks what is still being used. This requires storing more information in memory.

Head First Java - Chapter 1 Notes

3. Java Code Structure

- Now we will see the code structure in Java. We have the very big thing in the start which is the source file that ends with extension .java and contains the Java source code.

Inside it, the first thing you will see is the class. Inside the class, you have methods (like functions), and inside each method, you have statements to implement logic.

4. Main Method

- Where the JVM starts running, the first thing it tries to find is the:

PUBLIC STATIC VOID MAIN (STRING[] ARGS) {}

Then it starts to run the code inside it.

So, any Java app must have at least one class and just one main method for each application.

5. Java Language Rules

- In Java, everything is written inside a class.
- We can say that the Java Compiler and the Java Virtual Machine (JVM) are both very important and they complete each other.
- The compiler checks around 99% of the issues at compile time, like:
  - o Using the wrong data types
  - o Trying to access private or restricted members
  - o Syntax errors
  - o Other clearly wrong things
- The JVM checks at runtime, especially things not detected during compilation:
  - o Type casting issues (e.g., ClassCastException)
  - o Array bounds errors
  - o Dynamic behavior (e.g., loading new classes at runtime)

The JVM is more flexible than the compiler, which allows Java to support dynamic binding.

Dynamic binding means some objects or method calls might not be fully known at compile time but are handled at runtime (polymorphism or reflection).

Also, the JVM verifies and protects the bytecode before running it, to make sure no one has changed or hacked it.

Head First Java - Chapter 1 Notes

6. Extra Notes

- Java is an object-oriented language, which means everything must be in a class and everything is treated as an object.

- In Java, you cannot use integers directly as a boolean test in loops (e.g., while(x) is not valid even if x = 1). Only boolean values are allowed.

- Java includes a random library that you can use to generate random items from a list you specify.