Hello World!

In this lesson, you will get acquainted with the Hello World program and a basic introduction to Java.

WE'LL COVER THE FOLLOWING

Hello World: Example

The first program that most aspiring programmers write is the classic **“Hello World”** program. The *purpose* of this program is to display the text **“Hello World!”** to the user.

The **“Hello World”** example is somewhat famous as it is often the first program presented when introducing a programming language.

Try running the code below!

```
class HelloWorld {
    public static void main(String args[]) {
        System.out.println("Hello World!");
    }
}
```

> **Did you know?**
>
> The `System.out.println()` displays text in the current line of the console and then move the cursor to the beginning of the next line.

`Java` as a structural language

Before discussing the particulars, it is useful to think of a computer program in terms of both its **structure** and its **meaning** simultaneously.

A `Java` program is structured in a specific and particular manner. `Java` is a language and therefore has grammar similar to a spoken language like *English*. The grammar of computer languages is usually much, much simpler than spoken languages but comes with the disadvantage of having stricter rules. Applying this structure or grammar to the language is what allows the computer to understand the program and what it is supposed to do.

The overall program has a **structure**, but it is also important to understand the purpose of part of that **structure**. By analogy, a textbook can be split into sections, chapters, paragraphs, sentences, and words (the structure), but it is also necessary to understand the overall meaning of the words, the sentences, and chapters to fully understand the content of the textbook. You can think of this as the semantics of the program.

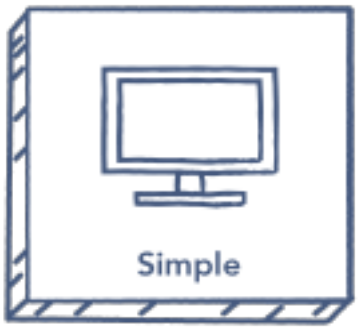
A line-by-line analysis of the program should give a better idea of both the **structure** and meaning of the classic **"Hello World"** program. Let's take a look at it in the upcoming lessons.

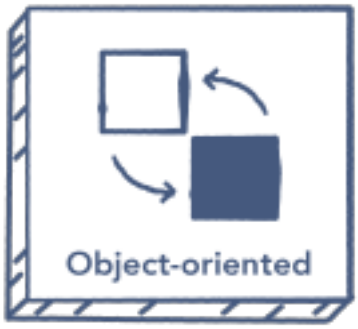
New features

The new features and upgrades included in Java changed the face of the programming environment and gave a new definition to *Object-Oriented Programming* (OOP in short). But unlike its predecessors, Java needed to be bundled with standard functionality and be independent of the host platform.

The primary goals in the creation of the Java language:

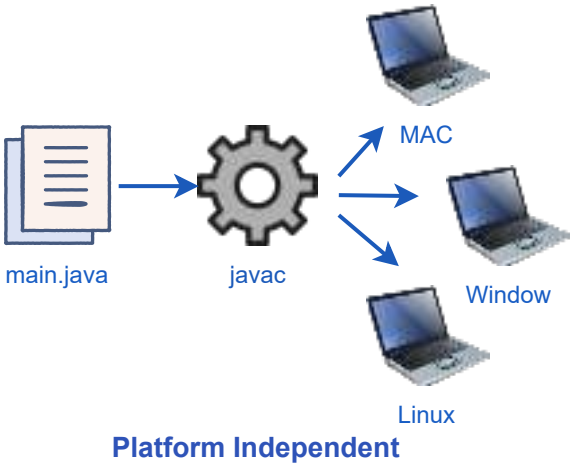
- It is **Simple & Portable**: Memory Management using Pointers is not allowed.





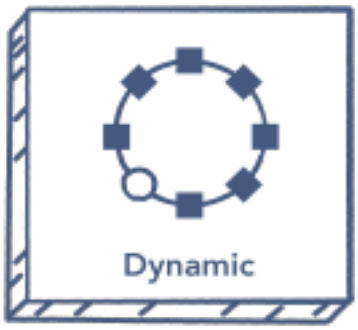
- It is **Object-Oriented**.

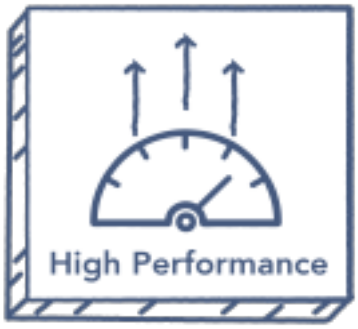
- It is **Independent** of the host platform.



- It is **Secured & Dynamic**: Designed to execute code from remote sources securely.

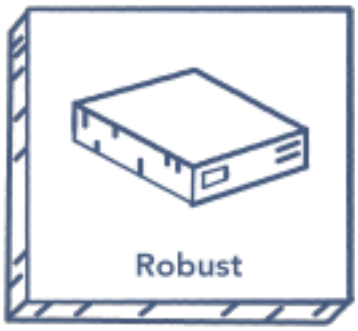
- It contains language facilities and libraries for **Networking.**

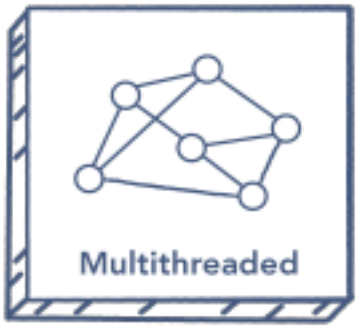




- **High Performance:** With the use of JIT (*Just-In-Time*) compilers, Java achieves high performance through the use of **byte-code** that can be easily translated into native machine code.

- It is **Robust**: Java has its own strong **memory management** system. This helps to eliminate errors as it checks the code during compile and runtime.





- Java is **Multithreaded:** It supports multiple executions of threads (i.e., *lightweight processes*), including a set of synchronization primitives.

The Java language introduces some new features that did not exist in other

The Java language introduces some new features that did not exist in other languages like C and C++.

Bad practices

Over the years, some features in C/C++ programming became abused by the programmers. Although the language allows it, it was known as bad practices. So the creators of Java have disabled them:

- Use of Pointers
- Operator overloading
- Multiple inheritance
- Friend classes (access another object's private members)
- Restrictions of explicit type casting (related to memory management)

Let's give you a detailed breakdown of the above code in the next lesson!