

# Módulo 4

## Introducción a Java

# ¿Cómo funciona un CPU?

Recordemos que los CPUs **sólo comprenden 1's y 0's**.

Internamente, el hardware de los CPUs contiene una tabla de instrucciones que puede realizar.

Cada vez que el CPU recibe un grupo de bits, se detiene, verifica qué operación está solicitando ejecutar, y la lleva a cabo.

# CPU Opcode Table

Opcode	Operación	Descripción
0x05	ADD	Add
0x20	AND	Logical AND operation
0xA0	MOV	Move
0X08	OR	Logical OR operation
0xF6	NOT	Logical NOT operation

Imaginemos que un CPU recibe la siguiente instrucción:  
**0x05A103**

Esta se divide en dos partes:

Opcode = Operation code

Argument = Los argumentos para realizar la operación

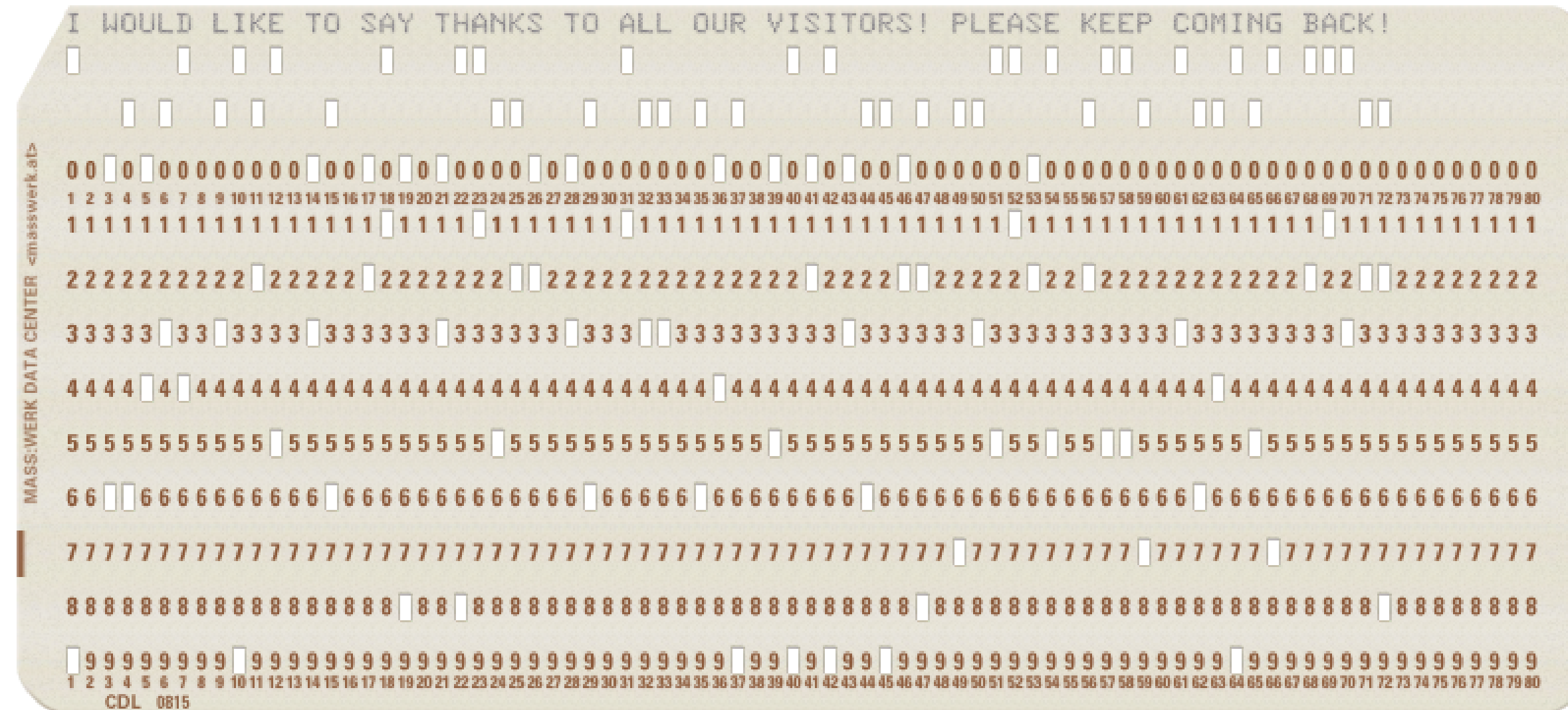
0x 05 A103  
    └──┬──┘ └──┬──┘  
      OPC ARG

# Video Playback Disabled

01	01110000	01110101	01100010	01101100	01101001	01100011	00100000	011000
02	01100001	01110011	01110011	00100000	01000110	01101001	01100010	011011
03	01100001	01100011	01100011	01101001	00100000	01111011	00001010	001000
04	00100000	00100000	01110000	01110101	01100010	01101100	01101001	011000
05	01110011	01110100	01100001	01110100	01101001	01100011	00100000	011101
06	01101001	01100100	00100000	01101101	01100001	01101001	01101110	001010
07	01110100	01110010	01101001	01101110	01100111	01011011	01011101	001000
08	01110010	01100111	01110011	00101001	00100000	01111011	00001010	001000
09	00100000	00100000	00100000	00100000	00100000	00100000	01101001	011011
10	00100000	01101110	00100000	00111101	00100000	00110001	00110000	001011
11	01110100	00110001	00100000	00111101	00100000	00110000	00101100	001000
12	00110010	00100000	00111101	00100000	00110001	00111011	00001010	001000
13	00100000	00100000	00100000	00100000	00100000	00100000	01010011	011110
14	01110100	01100101	01101101	00101110	01101111	01110101	01110100	001011
15	01110010	01101001	01101110	01110100	00101000	00100010	01000110	011010
16	01110011	01110100	00100000	00100010	00100000	00101011	00100000	011011
17	00101011	00100000	00100010	00100000	01110100	01100101	01110010	011011
18	00111010	00100000	00100010	00101001	00111011	00001010	00100000	001000
19	00100000	00100000	00100000	00100000	00100000	01100110	01101111	011100

Pudiéramos realizar todos nuestros programas de esta forma,  
pero sería muy tardado.

# Punch Cards



Tarjeta perforada utilizada en los 50's. Cada perforación indicaba una instrucción para la computadora.

# Punch Cards

Punch Card in  
Punch Card Machine



ComputerHope.com



# Punch Cards

Para facilitar y agilizar la programación, hemos inventado los...

# Lenguajes de Programación



# Lenguaje de Programación

Un lenguaje de programación es un **conjunto de vocabularios y reglas gramaticales** para dar instrucciones a una computadora.

Hay muchos lenguajes de programación, por ejemplo:

- C
- Java
- Python
- COBOL
- Pascal
- Javascript
- etc!

Los lenguajes de programación clasificamos dependiendo de qué tanto se parecen al **lenguaje natural**.

# Low-level Languages

- Altamente atados a un cierto procesador o pieza de hardware.  
¡Baja portabilidad!
- El procesador lo entiende directamente, por lo que la **ejecución es directa y rápida**.
- Son muy **difíciles de entender** o de mantener.
- Resolver errores o bugs requiere **mucho conocimiento técnico** (a veces hasta conocimiento de hardware!).
- Operarlos requiere habilidades muy específicas y **mucha experiencia**.

```
01 Circle
02
03 cseg segment
04         assume cs:cseg, ds:cseg, ss:cseg
05         org 100h
06         .386
07 start:
08
09         mov ax, 13h
10         int 10h
11
12         mov dx, 3c8h
13         xor al, al
14         out dx, al
15         inc dx
16         mov cx, 256
17         xor al, al
18 loopp:  out dx, al
19         out dx, al
20         out dx, al
21         inc al
22         dec cx
23         jnz loopp
24
25         mov ax, 0a000h
```

# High-level languages

- Diseñados para que los programas puedan ser ejecutados en distintas plataformas y arquitecturas. **Alta portabilidad**
- Estos lenguajes contienen muchas palabras en inglés, y generalmente son **fáciles de entender** por las personas.
- La ejecución de estos programas requiere que la computadora **convierta e interprete el código**.
- La mayoría del esfuerzo se puede dedicar al **diseño del programa**, pues tienen muchas herramientas que nos ayudan a manejar memoria, procesamiento, cálculos, etc.

```
01 public class Fibonacci {
02     public static void main(String[] args) {
03         int n = 10, t1 = 0, t2 = 1;
04         System.out.print("First " + n + " terms: ");
05         for (int i = 1; i <= n; ++i)
06         {
07             System.out.print(t1 + " + ");
08             int sum = t1 + t2;
09             t1 = t2;
10             t2 = sum;
11         }
12     }
13 }
```

Ejemplo código en un lenguaje de programación de alto nivel.

# ¿Cuál es mejor?

Los lenguajes de programación de **bajo nivel** permiten expresar **mayor capacidad al hardware**, pero la curva de aprendizaje y el **esfuerzo es considerablemente mayor**.








Los lenguajes de programación de **alto nivel** son **menos eficientes** en el uso de hardware, pero **reducen importantemente la complejidad de construcción de programas**.





# ¿Qué es Java?

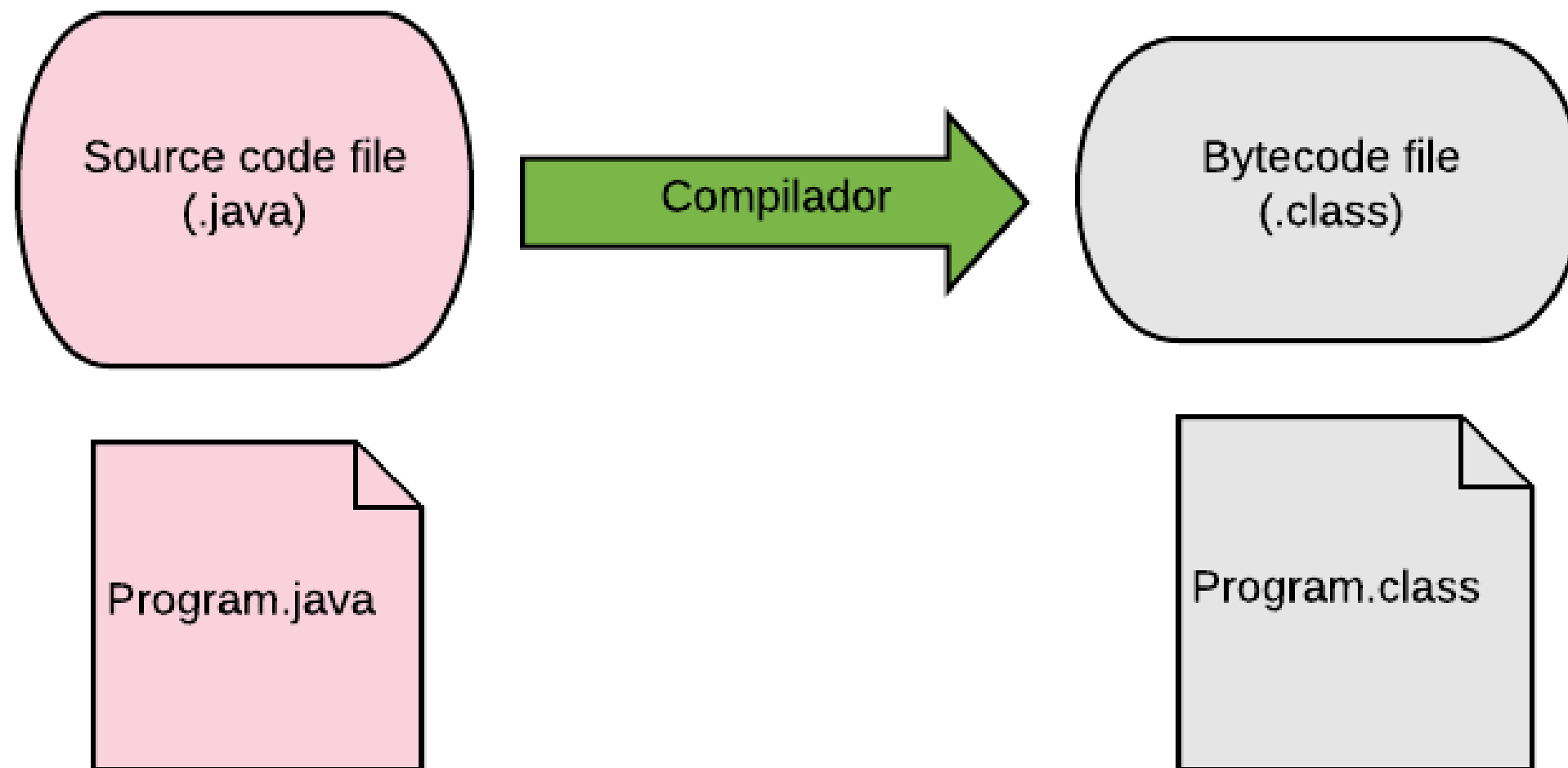
Es un lenguaje de programación de **alto nivel** que se ejecuta en una JVM (Java Virtual Machine). Sus principales fortalezas son:

- Cross-platform  
- Permite desarrollar aplicaciones móviles, web, videojuegos y software de servidor.  
- Orientado a Objetos 
- Muchas similitudes sintácticas con otros lenguajes de programación 
- Excelente primer lenguaje de programación 



Los programas de Java se llaman clases. Para poder ejecutar una clase, antes debemos **compilarla**. Este proceso se lleva a cabo mediante un compilador.






El compilador genera un archivo de bytecode.



# Compilador

El compilador es un programa que se encarga de convertir un archivo de código fuente (**source code**) en un archivo de código objeto (**object code**).

Para realizar esta traducción, el compilador

1. Lee los archivos de código fuente. 
2. Realiza verificaciones léxicas, sintácticas y semánticas.   
3. Genera los archivos de código objeto. 

# ¿Cómo funciona Java?

## 1. SOURCE CODE

El programador genera un archivo de código fuente (source code)

## 2. COMPILACIÓN

El compilador revisa la sintaxis del código. Si es correcto, genera un archivo de bytecode. Este archivo tiene el formato .class

## 3. BYTECODE

La Java Virtual Machine carga y verifica el bytecode.

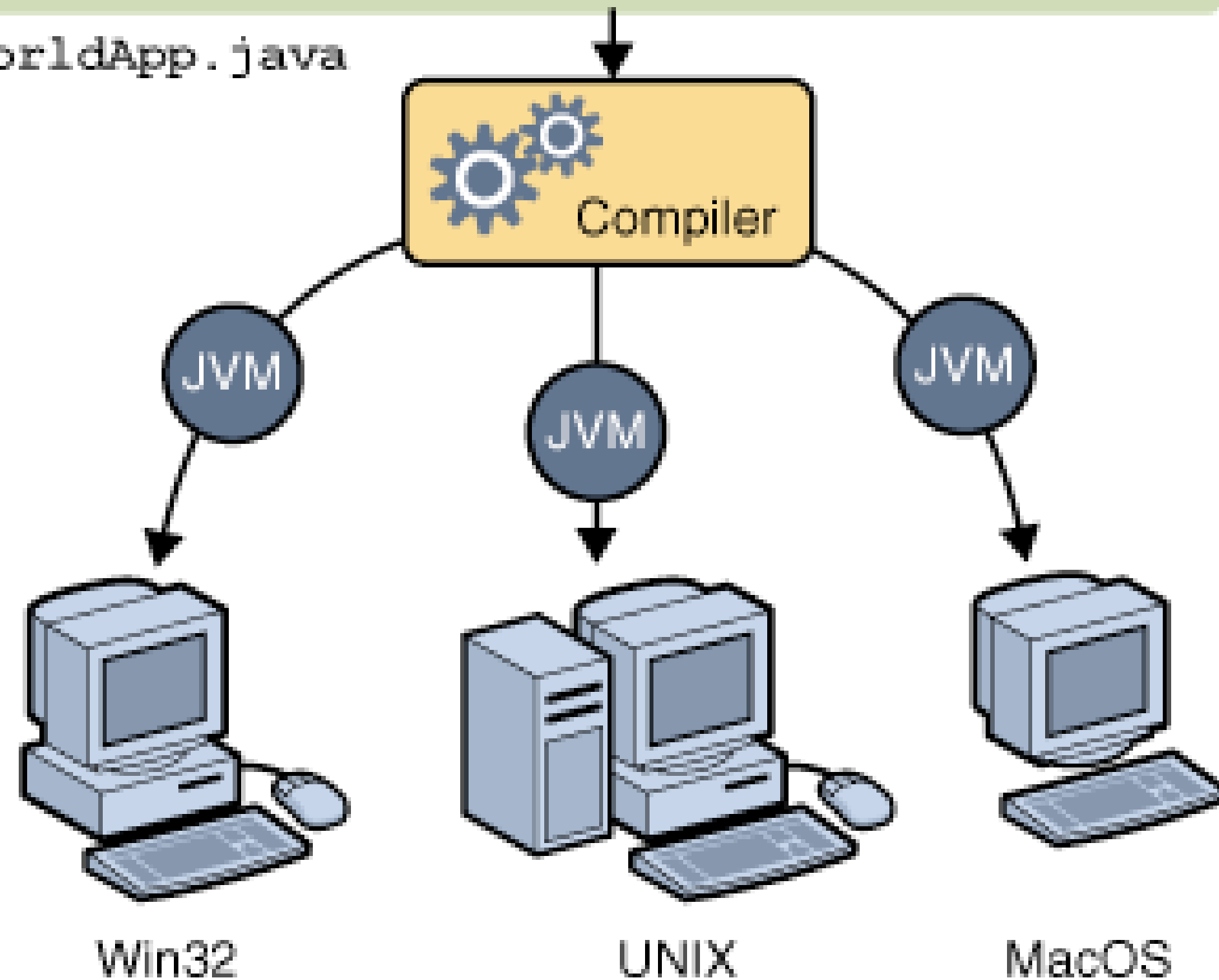
## 4. EXECUTION ENGINE

El Execution Engine carga las clases necesarias y ejecuta las instrucciones utilizando Just in Time compiling.

## Java Program

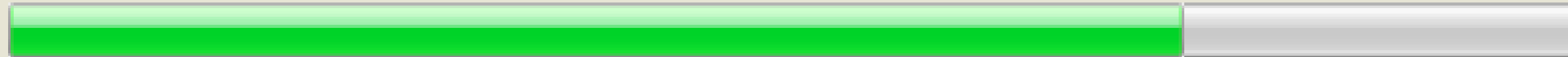
```
class HelloWorldApp {  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```

HelloWorldApp.java





Status: Installing Java



**3 Billion**  
**Devices Run Java**



**#1 Development Platform**

**ORACLE®**



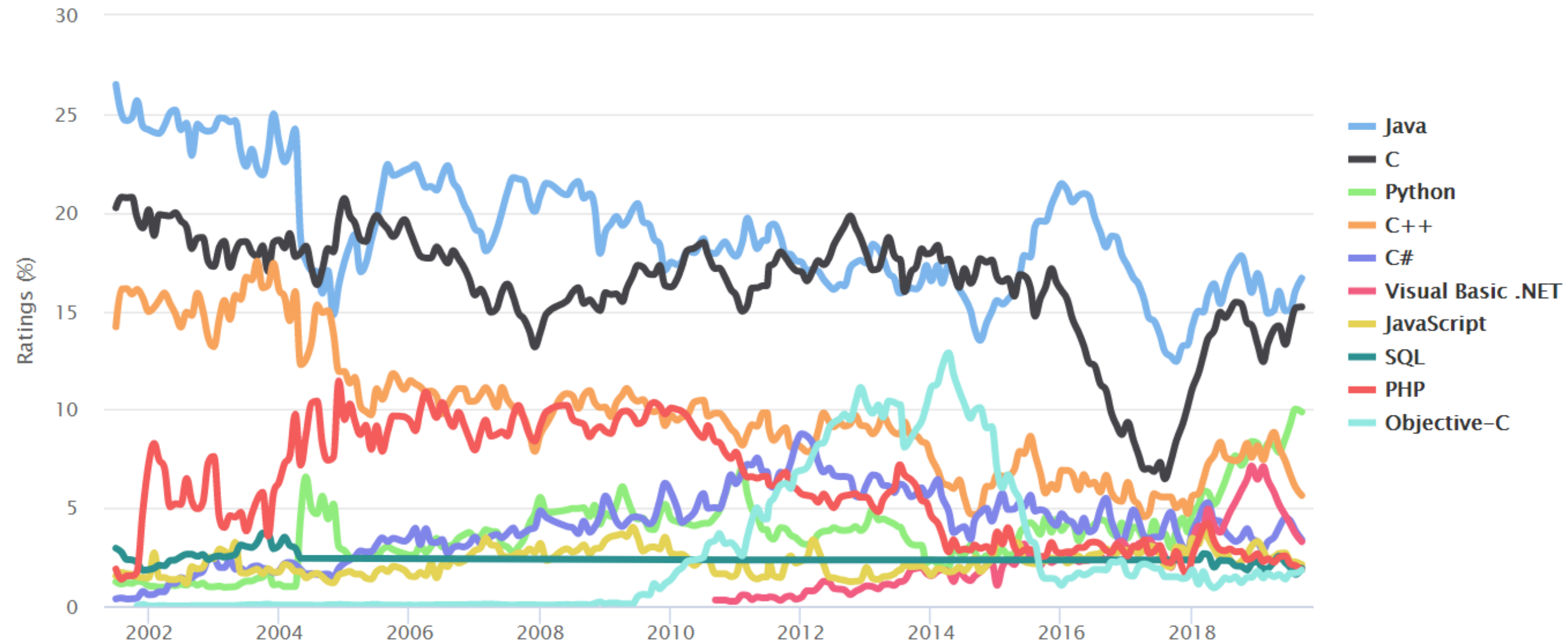
# Java Virtual Machine

Java funciona sobre un ambiente de ejecución que es independiente de la plataforma, llamado **Java Virtual Machine (JVM)**. Durante la ejecución, este ambiente está cargado en la RAM. Esta tiene tres funciones:

1. Extraer el bytecode (.class) e interpretarlo según la plataforma que lo está ejecutando (Android, Windows, Mac, Linux, celular)
2. Asegurarse que el bytecode es seguro.
3. ¡Ejecutar los programas!

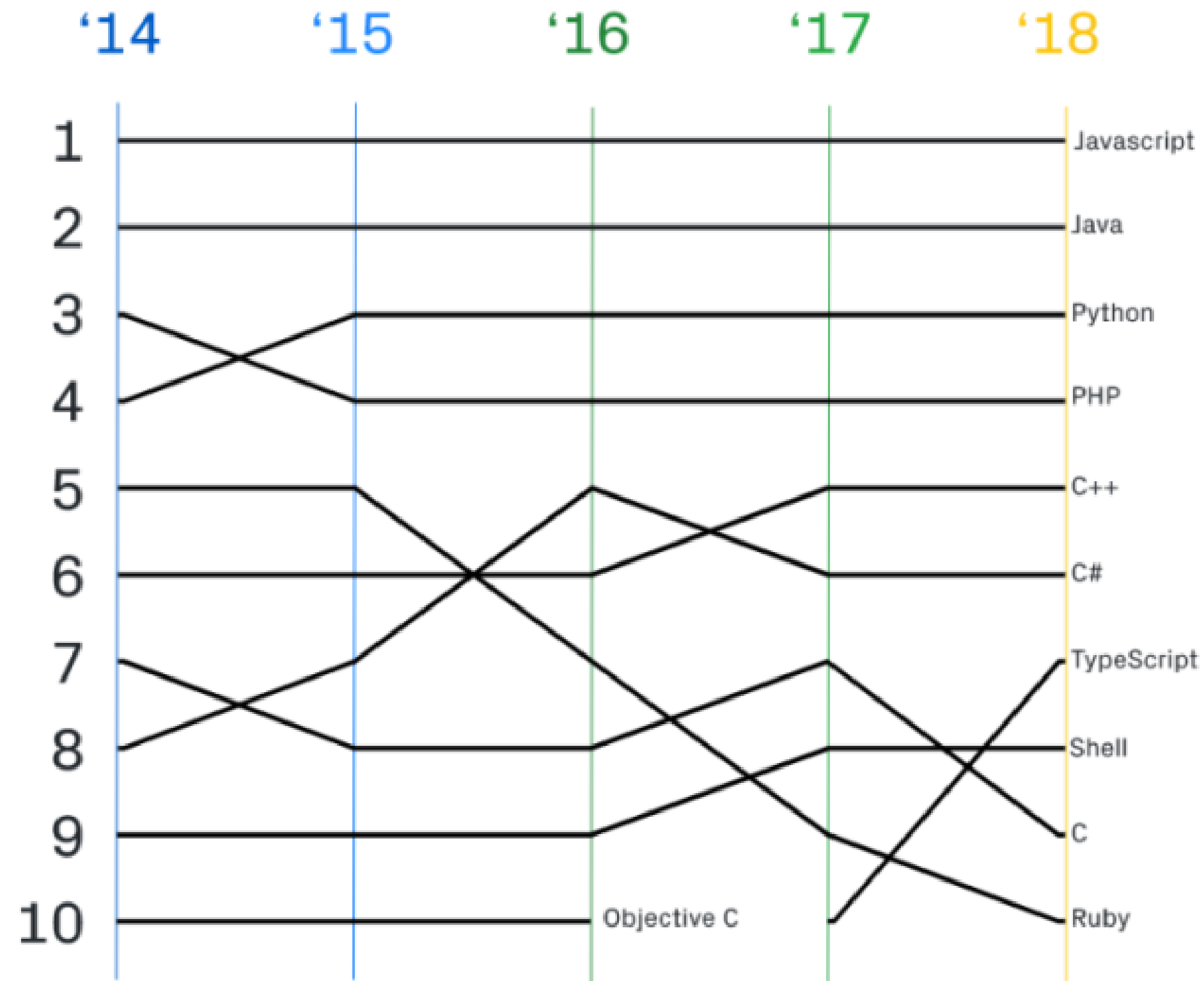
# TIOBE Rating

Source: [www.tiobe.com](http://www.tiobe.com)



Source

# Github's Octoverse Rating



Source

# Antes de comenzar...

- El código **no puede tener errores de ortografía o de sintaxis**. **"El perro salió a komer"** . ¡Nosotros lo entendemos, la computadora no!
- MAYÚSCULAS y minúsculas **hacen diferencia**.
- La indentación y los espacios son flexibles.

The only way to learn a new programming language is by writing programs in it. -  
Dennis Ritchie

# Hello, World!

```
01 public class HelloWorld {  
02  
03     public static void main(String[] args){  
04  
05         System.out.println("Hello, World!");  
06  
07     }  
08  
09 }
```

```
01 public class HelloWorld {  
02  
03     public static void main(String[] args){  
04  
05         System.out.println("Hello, World!");  
06  
07     }  
08  
09 }
```

El nombre de la clase es **HelloWorld**. Es importante que el nombre del archivo generado de java se llame igual que la clase + .java. En este caso, debería ser **HelloWorld.java**



```
01 public class HelloWorld {  
02  
03     public static void main(String[] args){  
04  
05         System.out.println("Hello, World!");  
06  
07     }  
08  
09 }
```

Seguido del nombre de la clase vienen las llaves { } Cada llave abierta implica que debe cerrarse más adelante la llave

```
01 public class HelloWorld {  
02  
03     public static void main(String[] args){  
04  
05         System.out.println("Hello, World!");  
06  
07     }  
08  
09 }
```

Se puede observar la declaración del método main. Todo lo que esté dentro de esta sección, dentro de las llaves, es el código del programa.

```
01 public class HelloWorld {  
02  
03     public static void main(String[] args){  
04  
05         System.out.println("Hello, World!");  
06  
07     }  
08  
09 }
```

Esta sentencia indica la llamada del método **println** de la clase **System.out**. Adentro podemos ver el **String** que se desplegará al ejecutar el programa: **Hello, World!**

```
01 public class HelloWorld {  
02  
03     public static void main(String[] args){  
04  
05         System.out.println("Hello, World!");  
06  
07     }  
08  
09 }
```