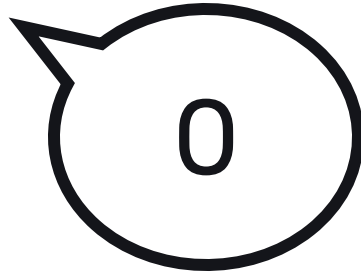
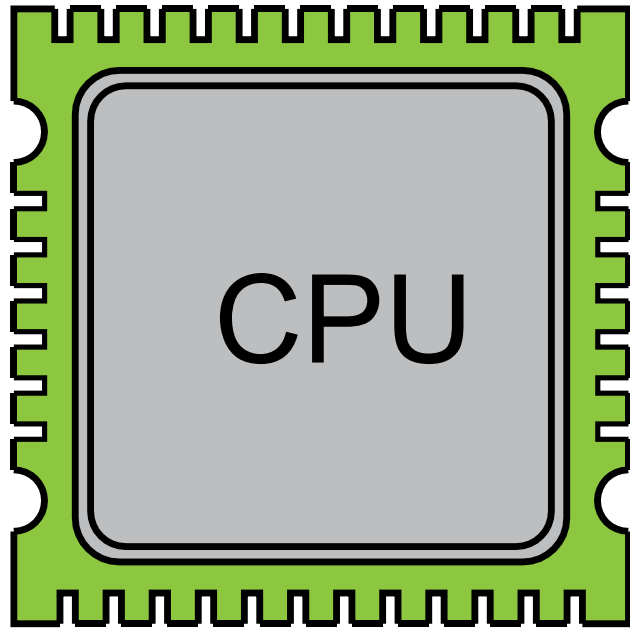
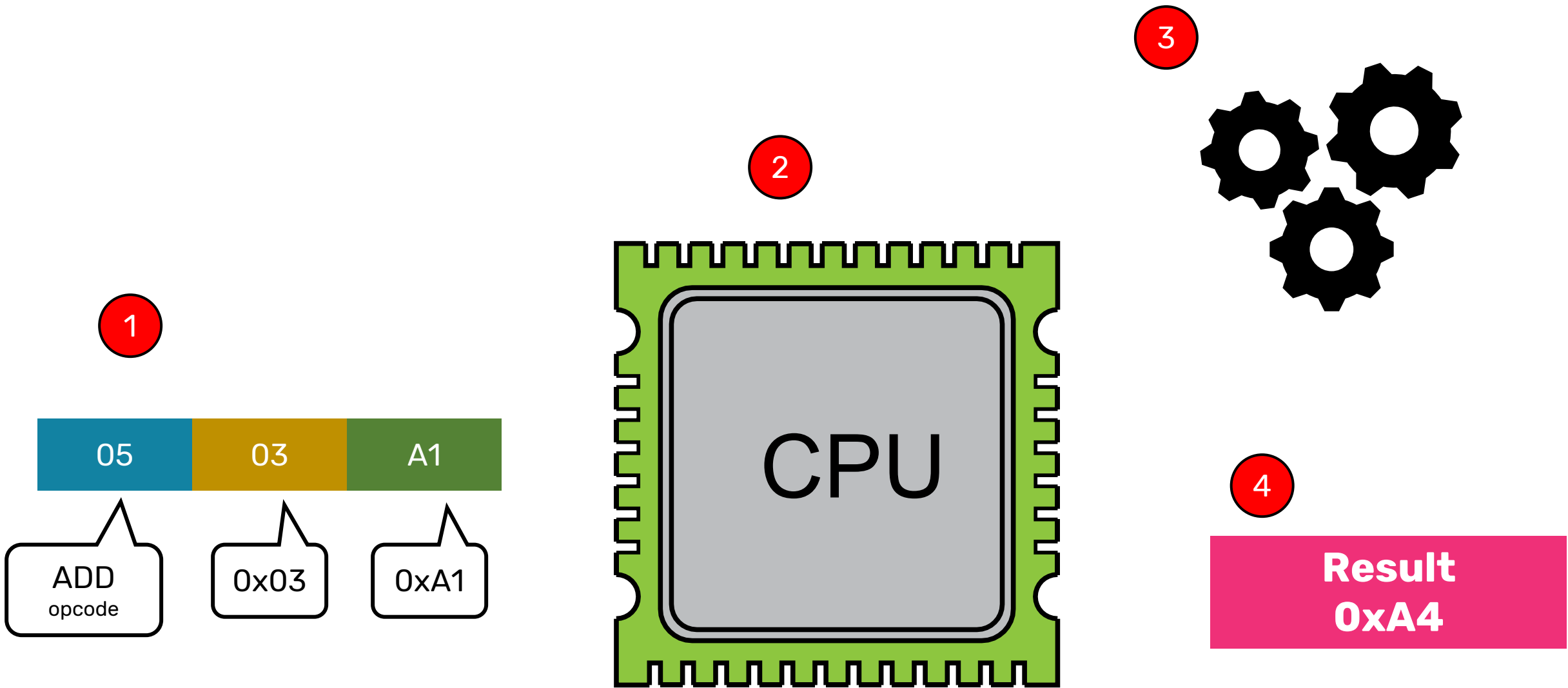


# Módulo 4

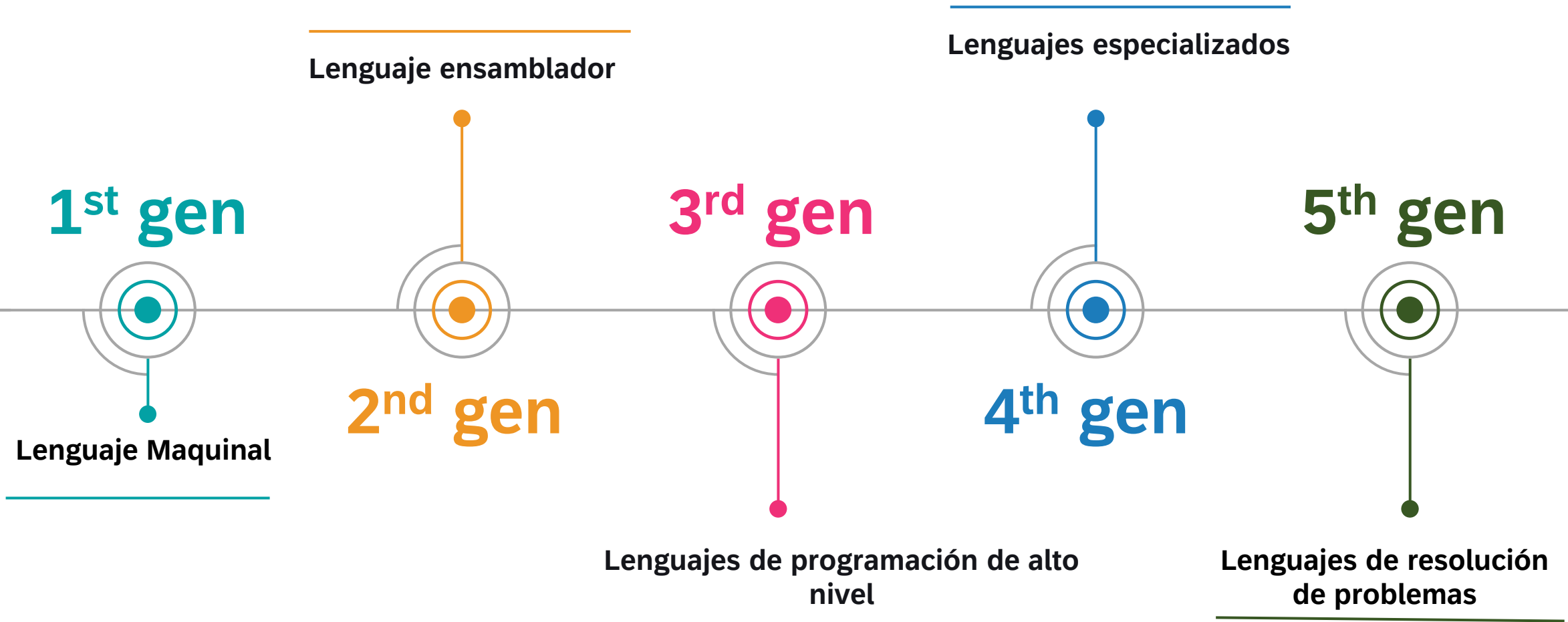
## Lenguajes de Programación



Recordemos que las  
computadoras  
entienden únicamente  
0's y 1's.



# Generaciones de Lenguajes de Programación

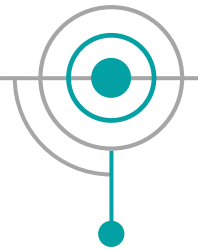


# 1GL (First Generation Language)

- 💡 Lenguaje maquina (machine language)
- 💡 Toda la codificación se hace en binario. El código no es legible para un ser humano.
- 💡 Nulo nivel de abstracción.
- 💡 El programador debe ser un experto en el software y el hardware en donde se ejecutará su programa.

```
0001001001000101  
0010010011101100  
10101101001...
```

1<sup>st</sup> gen

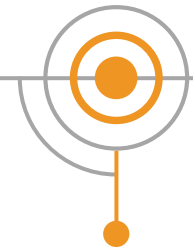


# 2GL (Second Generation Language)

- 💡 Lenguaje ensamblador (assembly language)
- 💡 Bajo nivel de abstracción.
- 💡 El código puede ser leído y escrito por programadores, pues las instrucciones se escriben utilizando mnemónicos (que sustituyen a los opcodes). Por ejemplo: LOAD, MUL, DIV, etc.

```
LOAD r1,b
LOAD r2,h
MUL r1,r2
DIV r1,#2
RET
```

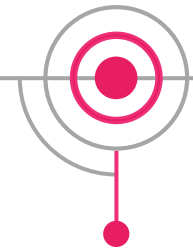
2<sup>nd</sup> gen



# 3GL (Third Generation Language)

- 💡 Lenguaje de programación de alto nivel (high-level programming language)
- 💡 Alto nivel de abstracción, con frases en inglés que sirven para representar instrucciones.
- 💡 Alta portabilidad (el mismo código se puede ejecutar en múltiples dispositivos).
- 💡 Java, C++, C#, Python

3<sup>th</sup> gen



```
1 public class Main {
2     private static final String GREETING_TEMPLATE = "Вітаю, пане %s";
3     private static final String NAME_UNKNOWN = "Невідомий";
4
5     public static void main(String[] args) {
6         if (args.length > 0) {
7             greeting(args[0]);
8         } else {
9             greeting();
10        }
11    }
12
13    private static void greeting() {
14        greeting(NAME_UNKNOWN);
15    }
16
17    /**
18     * Виводить рядок з привітанням (зазвичай, у консоль).
19     * @param name ім'я особи, до якої звернене привітання.
20     */
21    private static void greeting(String name) {
22        System.out.println(String.format(GREETING_TEMPLATE, name));
23    }
24 }
25
```

# 4GL (Fourth Generation Language)

- 💡 Lenguaje de dominio específico (domain specific programming language).
- 💡 Son lenguajes de programación especializados, con el objetivo de acelerar el diseño de programas complejos.
- 💡 Muy nivel de abstracción, con integración incluida a otros componentes (bases de datos, APIs, frameworks).
- 💡 SQL, ABAP, MATLAB, LabView

```
INSERT INTO v_alumnos1 (nombre, direccion, edad, estatus)
VALUES ('Bruno Díaz', 'Ciudad Gótica', 40, 'A');
SELECT * FROM alumnos;
UPDATE v_alumnos1 SET edad=35;
SELECT * FROM alumnos;
```

100 %

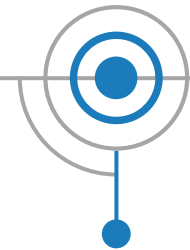
Results Messages

	id	nombre	direccion	edad	estatus
1	1	Bruno Díaz	Ciudad Gótica	40	A

	id	nombre	direccion	edad	estatus
1	1	Bruno Díaz	Ciudad Gótica	35	A

4<sup>th</sup> gen





# 5GL (Fifth Generation Language)

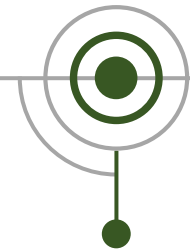
- Los 5GL son lenguajes de programación que resuelven problemas a partir de un set de reglas. Al definir las reglas, dichos lenguajes pueden diseñar un algoritmo que determine una solución.

```
salesman.pl
% initialise data, prepare graphics objects, and create the dialog

salesman :-
    tidy_salesman,
    init_salesman,
    Dstyle = [ws_caption,ws_maximizebox,ws_thickframe],
    Bstyle = [ws_child,ws_visible,ws_tabstop,bs_pushbutton],
    Sstyle = [ws_child,ws_visible,ss_left],
    Gstyle = [ws_child,ws_visible,ws_ex_clientedge],
    wcreate( salesman, 'Travelling Salesman', 10, 10, 520, 460,
    wcreate( (salesman,3), button, '&Exhaustive', 420, 8, 80, 22,
    wcreate( (salesman,4), button, '&Heuristic', 420, 38, 80, 22,
    wcreate( (salesman,5), button, '&Stop', 420, 68, 80, 22,
    wcreate( (salesman,6), button, '&Close', 420, 98, 80, 22,
    wcreate( (salesman,8), static, '', 10, 415, 480, 25,
    wcreate( (salesman,9), grafix, '', 10, 10, 400, 400,
    set_buttons( 0, 0, 0, 1 ),
    town_grafix,
    window_handler( salesman, salesman_handler ),
    call_dialog( salesman, _ ),
    tidy_salesman.

Prolog Source S C O R=481 C=45 L=26578 S=0
```

5<sup>th</sup> gen



**¿Qué es Java?**

Java Setup - Progress



Status: Installing Java



ATMs, Smartcards, POS Terminals, Blu-ray Players, PCs  
Set Top Boxes, Mobile Phones, Servers, Switches  
Routers, Smart Meters, Game Consoles, Medical Devices  
Automobiles, Park Meters, Lottery  
Systems, Access Control Systems, Building Controls  
Programs, and many more.

# 3 Billion Devices Run Java



Java™

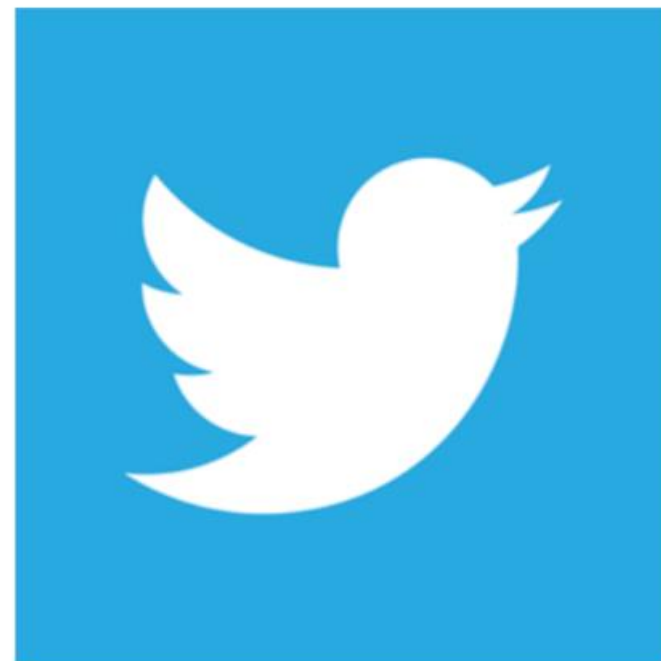
#1 Development Platform

ORACLE®

# ¿Qué es Java?

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

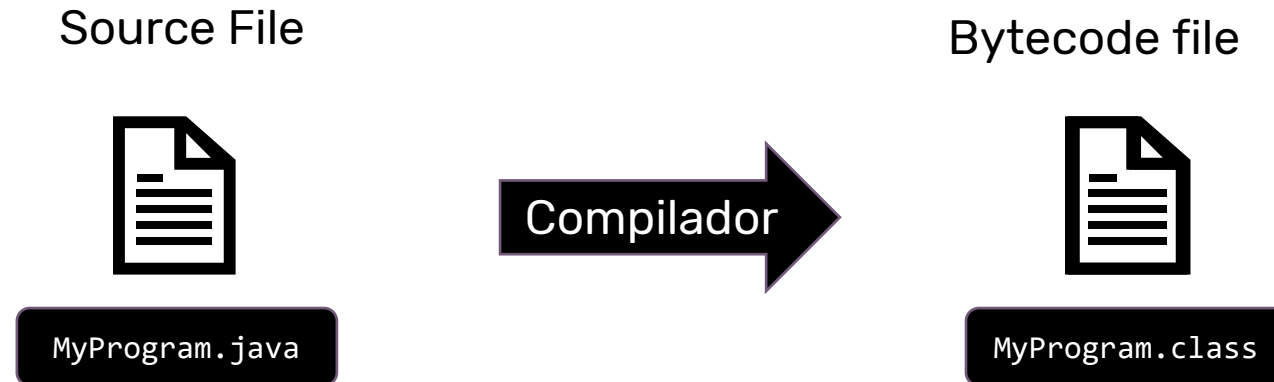
- **Cross-platform** (ejecutable en Windows, Mac, Linux, iOS, Android)
- Permite desarrollar **aplicaciones de cualquier tipo**: móviles, web, videojuegos, software de servidor, microservicios, APIs.
- Orientado a objetos
- Muchas **similitudes sintácticas con otros lenguajes de programación**.
- Excelente **primer** lenguaje de programación.



# Clases

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

El compilador de Java toma un archivo de **código fuente** (source code), y lo convierte en un archivo de código **bytecode**.



# Compilador

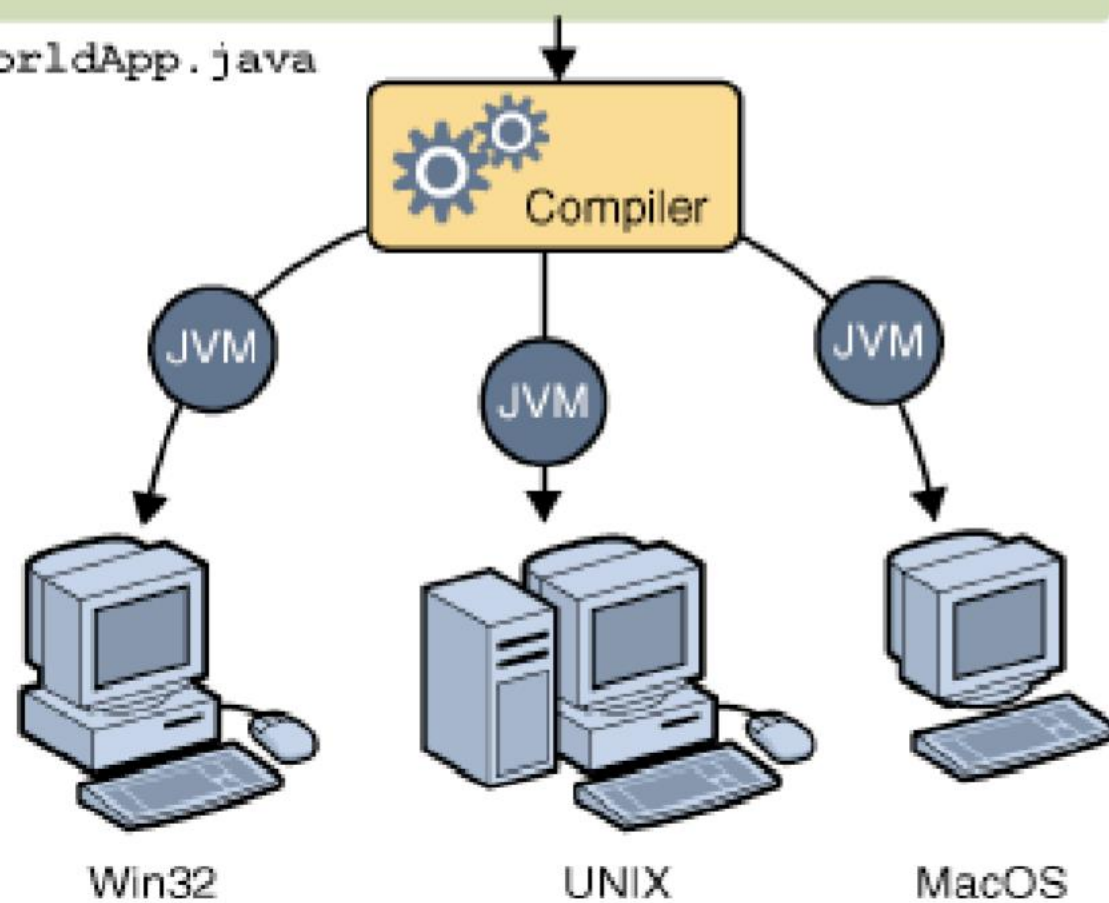
El compilador es un programa que se encarga de **traducir** nuestro código en algo que una computadora pueda entender. El compilador de Java cumple las siguientes funciones:

1. **Leer** los archivos de código fuente.
2. **Realizar verificaciones** léxicas, sintácticas y semánticas.
3. **Generar** los archivos de bytecode.

## Java Program

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

HelloWorldApp.java



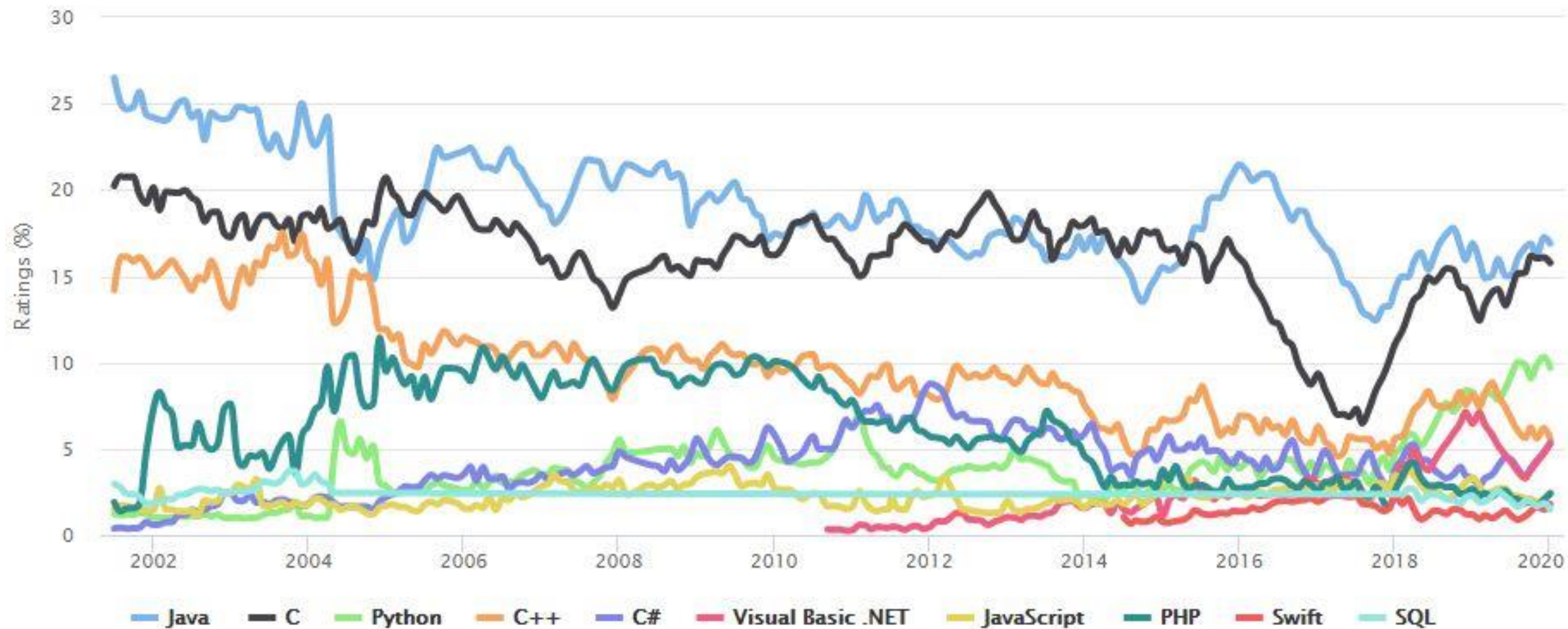


# Java Virtual Machine (JVM)

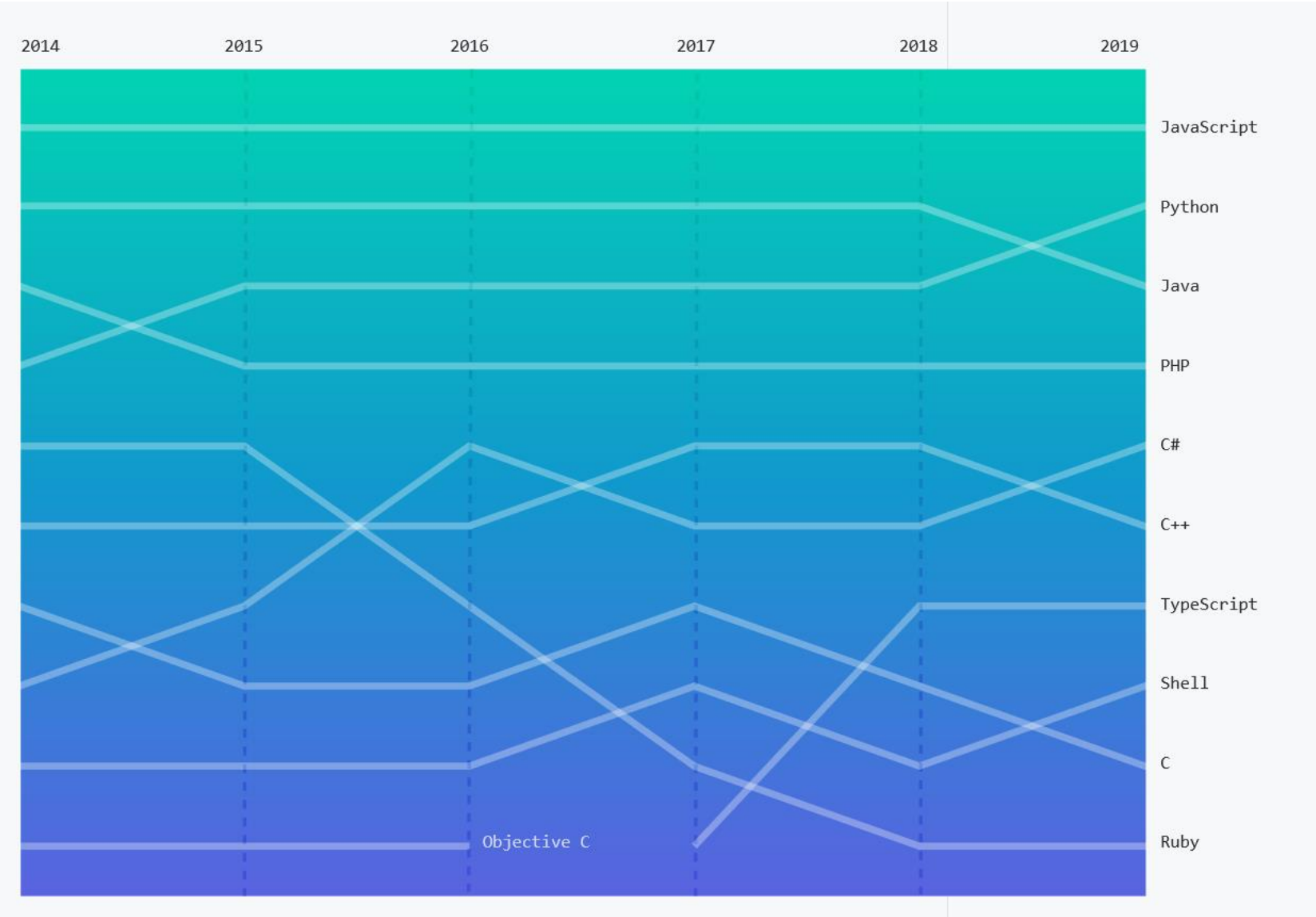
Java funciona sobre un ambiente de ejecución que es independiente de la plataforma, llamado Java Virtual Machine (JVM). Durante la ejecución, este ambiente está cargado en la RAM. Esta tiene tres funciones:

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



# Github's Octoverse Rating



# 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 **si 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

```
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 }
```