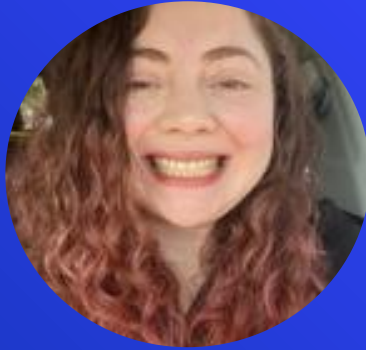


# Taller Introducción a Java



# Talleristas



**GRECIA SEPULVEDA**  
INGENIERA QUÍMICA



**BLANCA VERÓNICA ZÚÑIGA NÚÑEZ**  
MAESTRA EN CIENCIAS DE LA COMPUTACIÓN



# Temario

- ⬡ Introducción a Java
- ⬡ Conceptos básicos
- ⬡ Estructuras de Datos
- ⬡ Programación Orientada a Objetos
- ⬡ Desafío

# 1. Introducción a Java



# POR QUÉ JAVA



- Java es un lenguaje de programación veterano que ha perdurado desde su creación en 1995, manteniéndose como uno de los más influyentes y demandados en la industria tecnológica.





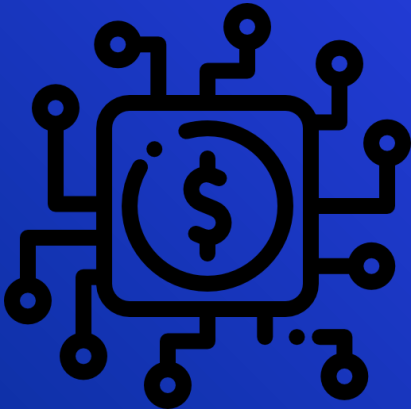
# POR QUÉ JAVA

- Es reconocido por su versatilidad y confiabilidad. Puedes desarrollar una amplia variedad de aplicaciones, incluyendo aplicaciones móviles, sistemas empresariales complejos y sitios web.



# POR QUÉ JAVA

- Java se destaca por su seguridad y robustez, características esenciales en aplicaciones críticas como sistemas bancarios, médicos y de control industrial.

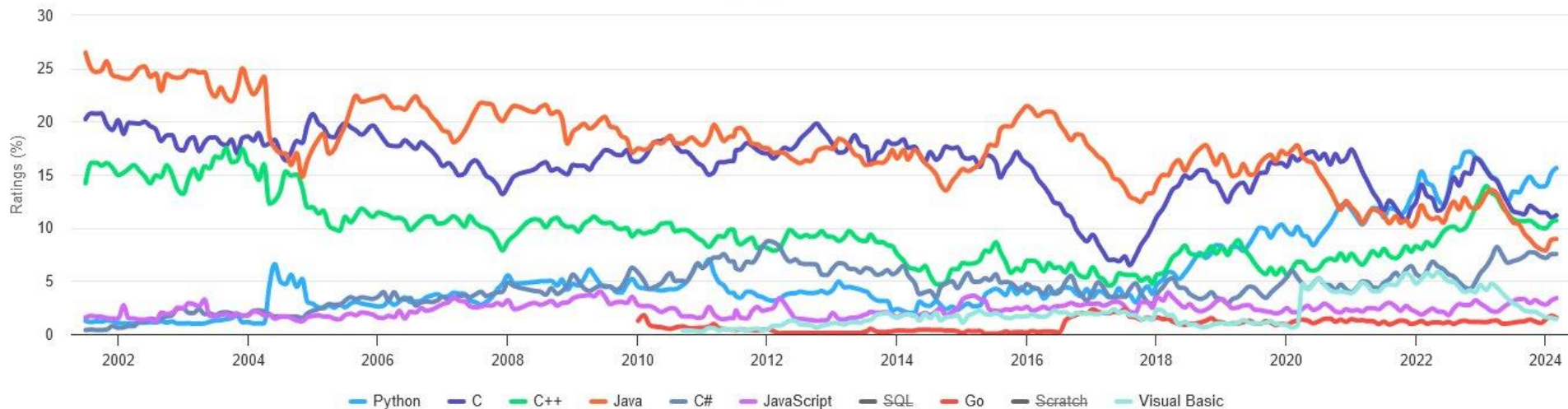


# Popularidad de Java



TIOBE Programming Community Index

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







# Rendimiento de Java

## mandelbrot

source	secs	mem	gz
<u>Python 3</u>	163.32	12,080	688
<u>Java</u>	4.15	69,136	796

## spectral-norm

source	secs	mem	gz
<u>Python 3</u>	120.99	13,424	407
<u>Java</u>	1.63	39,304	756

## n-body

source	secs	mem	gz
<u>Python 3</u>	567.56	8,076	1196
<u>Java</u>	6.74	35,844	1489


## 2. Conceptos básicos



# Conceptos básicos

## Variable

Espacios reservados en la memoria. Se reservan dándoles un nombre y un valor.



```
$currentMonth = "Febrero"
```

## Tipos de datos

Son los diferentes tipos de variables en las que se clasifica la información.



# Conceptos básicos

## PROGRAMAR

Actividad que está ligada a los procesos con los cuales se ejecutan tareas y programas en un dispositivo electrónico

## FUNCIÓN

Es un bloque de código reutilizable que realiza tareas específicas

## ALGORITMO

Son la secuencia de pasos lógicos que resuelven un problema, ¡son la base de la programación!



## 2. Fundamentos de Java



# TIPOS DE DATOS



**BYTE**

**INT**

**SHORT**

**LONG**

**FLOAT**

**DOUBLE**

**BOOLEAN**

# TIPOS DE VARIABLES



## LOCALES

se declaran en un método y solo son accesibles dentro de ese método



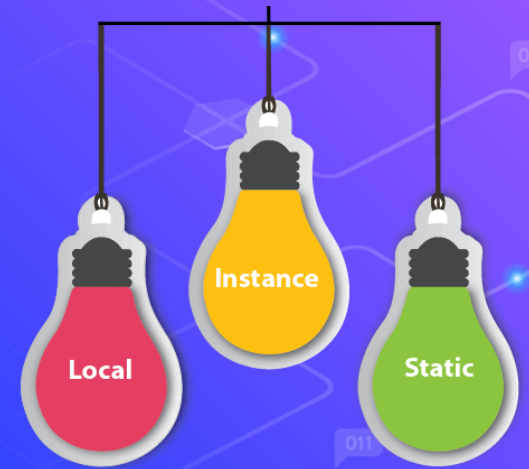
## DE INSTANCIA

pertenecen a una instancia específica de una clase y se declaran dentro de la clase pero fuera de cualquier método



## DE CLASE

son compartidas por todas las instancias de una clase y se declaran utilizando la palabra clave `static`



# NOMBRES DE VARIABLES

## VÁLIDAS

G

Nombre

\_nombre

Primer\_nombre

## INVÁLIDAS

int

primer nombre

n.o.m.b.r.e

2nombre





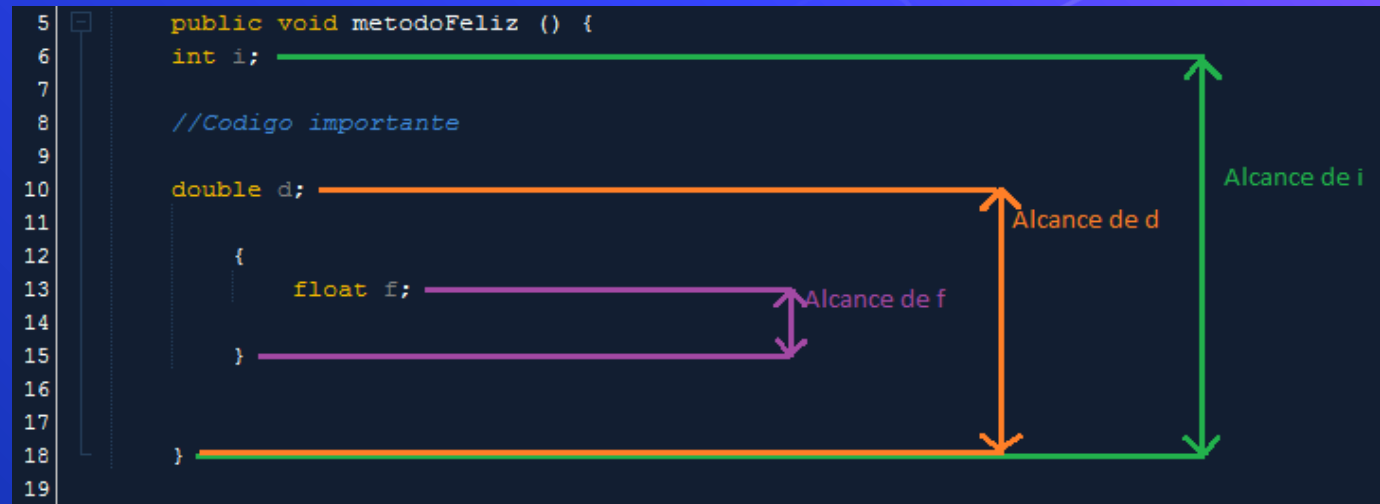
# ALCANCE

- En Java, el alcance o “scope” se refiere a la visibilidad y alcance de una variable, método o cualquier otro identificador en un programa

```

5 public void metodoFeliz () {
6   int i;
7
8   //Codigo importante
9
10  double d;
11
12  {
13    float f;
14  }
15
16
17
18 }
19

```





# OPERADOR DE ASIGNACIÓN

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
=	Operador asignación	n = 4	n vale 4

# OPERADORES ARITMÉTICOS

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
-	operador unario de cambio de signo	-4	-4
+	Suma	2.5 + 7.1	9.6
-	Resta	235.6 - 103.5	132.1
*	Producto	1.2 * 1.1	1.32
/	División (tanto entera como real)	0.050 / 0.2 7 / 2	0.25 3
%	Resto de la división entera	20 % 7	6

# OPERADORES ARITMÉTICOS INCREMENTALES

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
<b>++</b>	<b>Incremento</b> i++ primero se utiliza la variable y luego se incrementa su valor ++i primero se incrementa el valor de la variable y luego se utiliza	4++ a=5; b=a++; a=5; b=++a;	5  a vale 6 y b vale 5  a vale 6 y b vale 6
<b>--</b>	<b>decremento</b>	4--	3

# OPERADORES ARITMÉTICOS COMBINADOS

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
<b>+=</b>	Suma combinada	$a+=b$	$a=a+b$
<b>-=</b>	Resta combinada	$a-=b$	$a=a-b$
<b>*=</b>	Producto combinado	$a*=b$	$a=a*b$
<b>/=</b>	División combinada	$a/=b$	$a=a/b$
<b>%=</b>	Resto combinado	$a\%=b$	$a=a\%b$

# OPERADORES DE RELACIÓN

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
<code>==</code>	igual que	<code>7 == 38</code>	false
<code>!=</code>	distinto que	<code>'a' != 'k'</code>	true
<code>&lt;</code>	menor que	<code>'G' &lt; 'B'</code>	false
<code>&gt;</code>	mayor que	<code>'b' &gt; 'a'</code>	true
<code>&lt;=</code>	menor o igual que	<code>7.5 &lt;= 7.38</code>	false
<code>&gt;=</code>	mayor o igual que	<code>38 &gt;= 7</code>	true

# OPERADORES LÓGICOS

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
!	Negación - NOT (unario)	!false !(5==5)	true false
	Suma lógica – OR (binario)	true   false (5==5)   (5<4)	true true
^	Suma lógica exclusiva – XOR (binario)	true ^ false (5==5)   (5<4)	true true
&	Producto lógico – AND (binario)	true & false (5==5) & (5<4)	false false
	Suma lógica con cortocircuito: si el primer operando es true entonces el segundo se salta y el resultado es true	true    false (5==5)    (5<4)	true true
&&	Producto lógico con cortocircuito: si el primer operando es false entonces el segundo se salta y el resultado es false	false && true (5==5) && (5<4)	false false



# ESTRUCTURAS DE CONTROL

Las estructuras de control en java las emplearemos para modificar el flujo secuencial de un programa. Esto es, el orden en el que se van ejecutando las instrucciones de este.





# IF

- ⬡ La función evalúa una condición y ejecuta un bloque de instrucciones en el caso de que la condición sea verdadera.



# IF-ELSE

## IF- Else

```
/* Ejemplo Estructura IF - */  
if (condición) {  
    // Ejecuta si cumple condición  
    instrucciones  
} else {  
    // Si no se cumple la condición  
    instrucciones  
}
```



# BUCLES - for

When you know exactly how many times you want to loop through a block of code, use the *for* loop

```
1 | for(variable initialization; condition; code execution){  
2 |     //enter your code here that will be called repeatedly.  
3 | }
```



# BUCLES - while

The *while* loop loops through a block of code as long as a specified condition is true

```
1  while(condition) {  
2      //codes to be executed if condition  
3      //is true.  
4  }
```

# 2.

# Estructuras de Datos

Arrays Unidimensionales y  
Multidimensionales





# Arrays unidimensionales

- Permite almacenar un conjunto de elementos del mismo tipo bajo un único identificador.

myArray	10	15	8	9	0	10
Posiciones ->	0	1	2	3	4	5



# Arrays unidimensionales

- ⬡ Declaración
- ⬡ Creación
- ⬡ Inicialización
  - Al momento de creación
  - Posteriormente a la creación
- ⬡ Algunos métodos útiles



# Arrays multidimensionales

- Permite almacenar un conjunto de elementos del mismo tipo bajo un único identificador **en una matriz**.

myArray

20	15	8
9	0	10



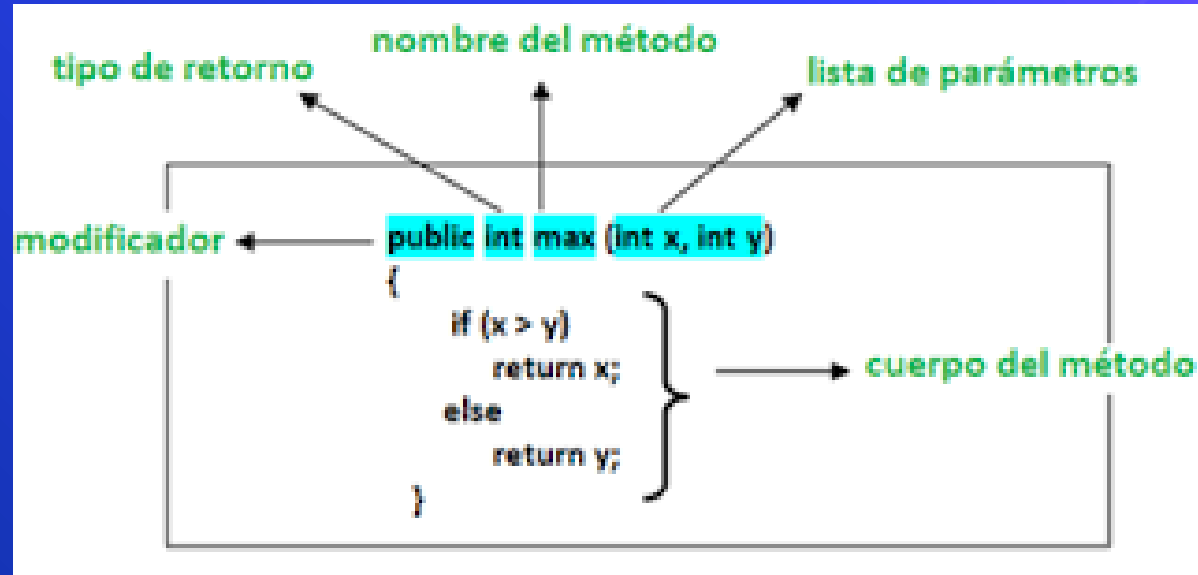


# Métodos

- ⬡ Son bloques de código que realizan una tarea específica y pueden ser invocados (o llamados) desde otras partes del programa.



# Métodos



# Métodos - void



- Los métodos pueden o no regresar un valor.



# 2. Programación Orientada a Objetos

OOP





# OOP

- ❡ Paradigma de programación que se basa en "objetos", que son entidades que combinan datos y operaciones que actúan sobre esos datos.



# OOP

- ⬡ Todo en java es un objeto, excepto los tipos primitivos:
  - byte, short, int, long, float, double, boolean y char.



# OOP

- Es una manera de crear código.
- Se intenta modelar a través de entidades u objetos.
- Fácil de reusar.
- Los objetos tienen atributos y funciones (métodos).



# Pilares de la POO

- ⬡ Abstracción
- ⬡ Herencia
- ⬡ Polimorfismo
- ⬡ Encapsulación





# Pilares de la POO

- ⬡ Abstracción
- ⬡ Herencia
- ⬡ Polimorfismo
- ⬡ Encapsulación

Ocultar información al usuario.



# Pilares de la POO

- ⬡ Abstracción
- ⬡ Herencia
- ⬡ Polimorfismo
- ⬡ Encapsulación

Heredar atributos y métodos.



# Clases

- Una clase es una plantilla o un modelo que define las propiedades y comportamientos (métodos) que los objetos de ese tipo pueden tener.



# Ejemplos

## ❖ Clase: Frutas

- Objetos: Manzana, plátano y mango.

## ❖ Clase: Perro

- Objetos: Chihuahua, Husky y Bulldog.

# Clases



## Persona

- Nombre
- Edad
- Nacionalidad
- + Saludar
- + Presentarse

# Campos (variables de instancia, atributos)



- Los campos (atributos) de una clase representan el estado o las características que los objetos de esa clase pueden tener.



# Métodos

- Los métodos de una clase representan el comportamiento o las acciones que los objetos de esa clase pueden realizar.





# Constructor

- Un constructor es un método especial que se utiliza para inicializar un objeto justo después de su creación.
- Debe coincidir con el nombre de la clase.
- Puede o no aceptar parámetros.

Constructor  
Predeterminado







# Estructura de una clase

- ⬡ Modificador de acceso
- ⬡ Palabra clave `class`
- ⬡ Nombre de la clase
- ⬡ Cuerpo de la clase
  - Variables y métodos





# Creación de objetos (instancia de clase)

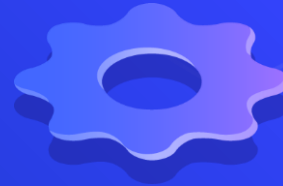
- ⬡ Nombre de la clase
- ⬡ Nombre del objeto
- ⬡ Palabra reservada new
- ⬡ Parámetros del constructor





# Palabra reservada this

- Se utiliza para hacer referencia a los campos, métodos y constructores de la instancia actual de la clase.





# Modificadores de acceso

## Public

- Accesible desde cualquier clase en cualquier paquete.

## Private

- Accesible únicamente desde la misma clase.

## Protected

- Accesible desde la misma clase, clases del mismo paquete y clases derivadas (subclases).



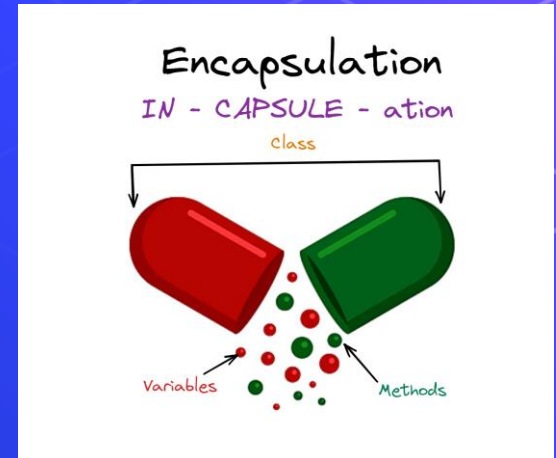
# Scope (alcance)

- Región del código en la que una variable, constante, función o clase es accesible y puede ser utilizada.

{ }

# Encapsulamiento

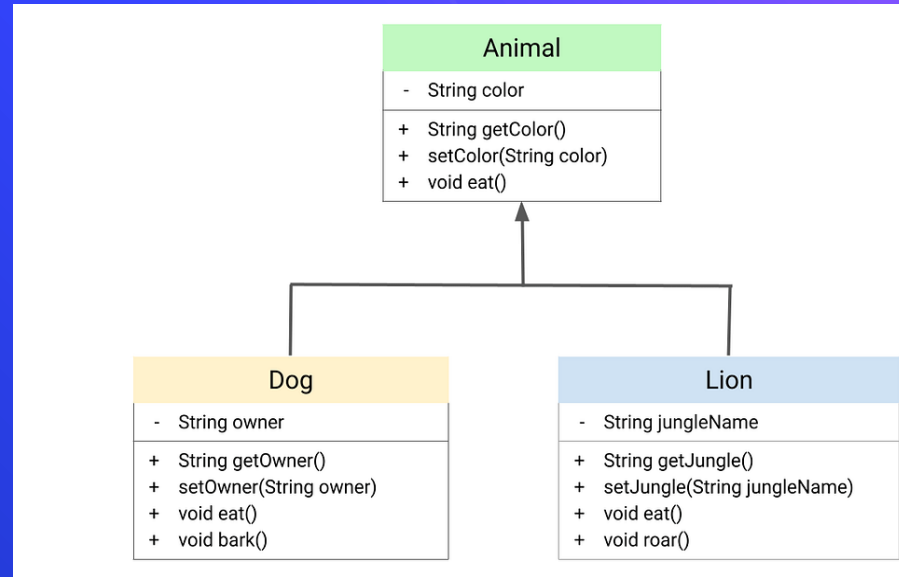
- Consiste en ocultar los detalles de implementación de una clase y exponer solo los métodos necesarios para interactuar con los objetos.





# Herencia

- Permite crear una nueva clase basada en una clase existente.
- “Heredando” atributos y métodos.





# Jueces automáticos

⬡ <https://omegaup.com/>

⬡ <https://www.hackerrank.com/>

**omegaUp**

**HackerRank**■