

# Unidad 1

## 01 - Lenguaje Java





# Historia de Java

Java nace en 1991 con el nombre "OAK", posteriormente cambiado por Green por problemas legales, y finalmente con la denominación actual JAVA.

El objetivo de java era crear un lenguaje de programación parecido a C++ en estructura y sintaxis, fuertemente orientado a objetos, pero con una máquina virtual propia. Esto se hizo bajo el principio, de poder ser usado bajo cualquier arquitectura "Write Once, Run Anywhere" (escribelo una vez, ejecútalo en cualquier sitio).

A día de hoy, podemos decir, que Java es uno de los lenguajes más importantes del mundo. Con una comunidad extendida en todos los componentes y más de 4 millones de desarrolladores, existen millones de dispositivos que lo usan.

Además, tras el surgimiento de android, Java se estableció como el lenguaje de programación para móviles más extendido del planeta.



# Versiones de Java

Versión	Release date	Public Updates	Support Until
JDK Beta	1995	?	?
JDK 1.0	January 1996	?	?
JDK 1.1	February 1997	?	?
J2SE 1.2	December 1998	?	?
J2SE 1.3	May 2000	?	?
J2SE 1.4	February 2002	October 2008	February 2013
J2SE 5.0	September 2004	November 2009	April 2015
Java SE 6	December 2006	April 2013	December 2018
Java SE 7	July 2011	April 2015	July 2022
Java SE 8 (LTS)	March 2014	At least May 2026 for OpenJDK	December 2030

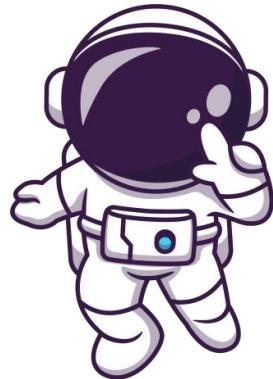
[https://en.wikipedia.org/wiki/Java\\_version\\_history](https://en.wikipedia.org/wiki/Java_version_history)



# Versiones de Java

Versión	Release date	Public Updates	Support Until
Java SE 9	September 2017	March 2018 for OpenJDK	N/A
Java SE 10	March 2018	September 2018 for OpenJDK	N/A
Java SE 11 (LTS)	September 2018	October 2024 for OpenJDK	September 2026
Java SE 12	March 2019	September 2019 for OpenJDK	N/A
Java SE 13	September 2019	March 2020 for OpenJDK	N/A
Java SE 14	March 2020	September 2020 for OpenJDK	N/A
Java SE 15	September 2020	March 2021 for OpenJDK	N/A
Java SE 16	March 2021	September 2021 for OpenJDK	N/A
Java SE 17 (LTS)	September 2021	September 2030 for Zulu	TBA

[https://en.wikipedia.org/wiki/Java\\_version\\_history](https://en.wikipedia.org/wiki/Java_version_history)



# Plataformas Java: Java ME, Java SE, Java EE





# The Java™ Platform



Optional Packages

Java 2  
Enterprise  
Edition  
(J2EE)

Optional Packages

Java 2  
Standard  
Edition  
(J2SE)

JVM

Java 2 Platform Micro Edition  
(J2ME™)

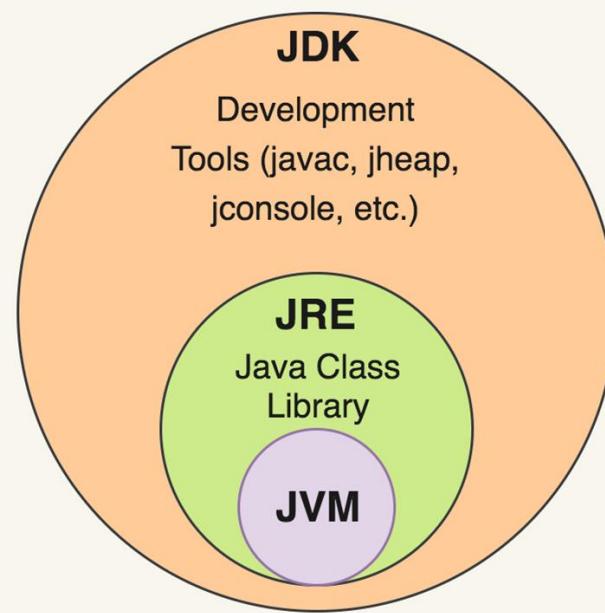


QU CONECTAN ✓



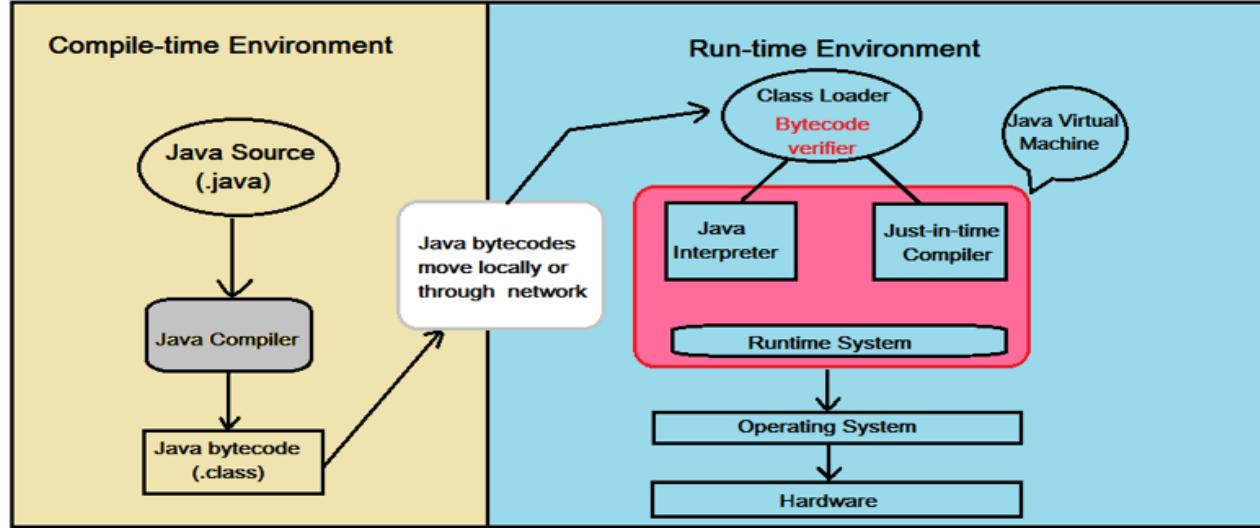
# JVM, JRE, JDK

- **Java Virtual Machine (JVM)**
  - Es una abstracción de una máquina de cómputo, encargada de ejecutar los programas Java.
- **Java Runtime Environment (JRE)**
  - Es un paquete de software que contiene los artefactos requeridos para ejecutar un programa Java.
- **Java Development Kit (JDK)**
  - Es un superconjunto del JRE y contiene las herramientas para los programadores Java.





# Compilado o interpretado?

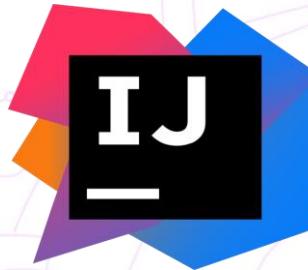




# Entorno de Desarrollo Integrado - IDE



Apache  
Netbeans



IntelliJ  
IDEA



Oracle  
JDeveloper



Visual Studio  
Code



Eclipse  
IDE

... y más  
**Hechos**  
QUE CONECTAN ✓



# Visual Studio Code

A screenshot of the Visual Studio Code interface. The title bar shows "blog-post.js - gatsby-graphql-app - Visual Studio Code - Insiders". The left sidebar displays the "EXTENSIONS MARKETPLACE" with various extensions listed, such as Python, GitLens, C/C++, ESLint, Debugger for Chrome, Language Support for Java, vscode-icons, Vetur, and C# for Visual Studio Code. The main editor area shows a portion of a JavaScript file named "blog-post.js" with code related to Gatsby and GraphQL. The bottom status bar indicates the file is "master", has 0 changes, and is in "JavaScript" mode.



<https://code.visualstudio.com/>





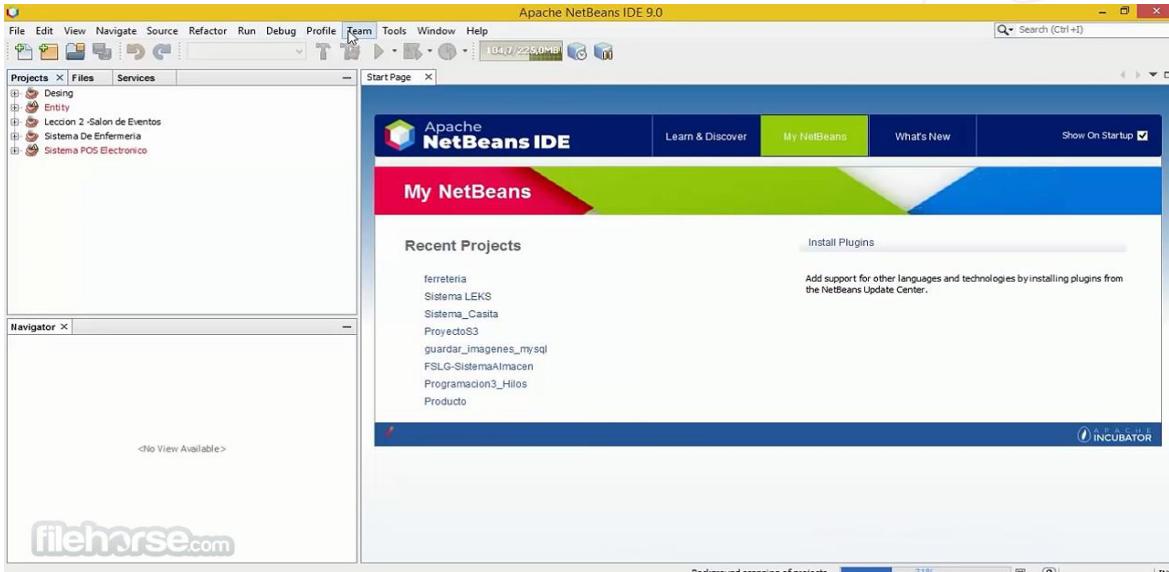
El futuro digital  
es de todos

MinTIC

# Netbeans IDE

UTP  
Universidad Tecnológica  
de Pereira

Misión  
TIC 2022



<https://netbeans.apache.org/>





El futuro digital  
es de todos

MinTIC

UTP  
Universidad Tecnológica  
de Pereira

Misión  
TIC 2022

# Preparando el ambiente para el ciclo

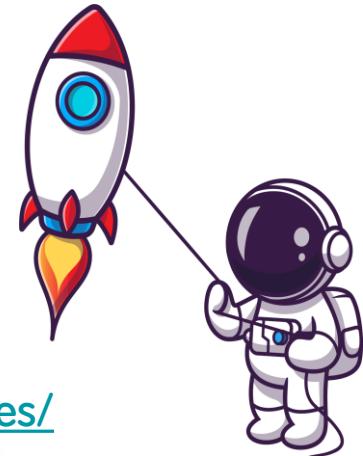


Hechos  
QUE CONECTAN ✓



# Herramientas a instalar

- Visual Studio Code - <https://code.visualstudio.com/download>
- OpenJDK 11 (LTS)
  - AdoptOpenJDK - <https://adoptopenjdk.net/>
- Git - <https://git-scm.com/downloads>
- Maven - <https://maven.apache.org/download.cgi>
- Bases de Datos
  - SQLite - <https://www.sqlite.org/download.html>
  - DBeaver Community - <https://dbeaver.io/download/>
- UML
  - StartUML - Herramienta Case - <https://staruml.io/download>
  - PlantUML - Diagramas como Código - <https://plantuml.com/es/>
  - Diagrams.Net - Online Diagram - <https://app.diagrams.net/>





El futuro digital  
es de todos

MinTIC

UTP  
Universidad Tecnológica  
de Pereira

Misión  
TIC 2022

# Un recorrido por el lenguaje



Hechos  
QUE CONECTAN ✓



# Comentarios

// Este es un comentario de línea

```
/*
```

Este es un comentario de bloque

Todo entre estos símbolos es ignorado

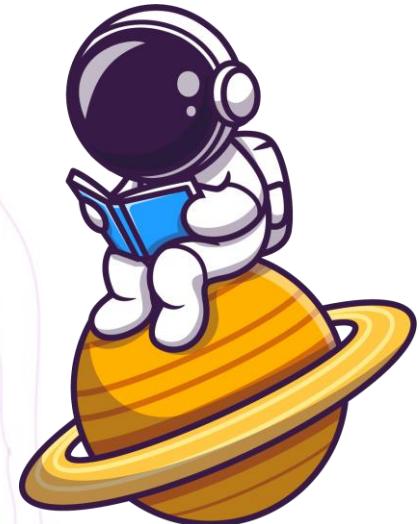
```
*/
```

```
/**
```

\* Este es un comentario de documentación.

\* Se usa para el Javadoc de nuestro proyecto

```
*/
```





# Sentencias



- Una sentencia es una orden que se le da al programa para realizar una tarea específica.
- Las sentencias acaban con punto y coma (;). Este carácter separa una sentencia de la siguiente.

```
int i=1;  
import java.awt.*;  
System.out.println("El primer programa");  
rect.moveTo(10, 20);
```

- Los caracteres espacio en blanco se pueden emplear libremente. Son importantes para la legibilidad de un programa, la colocación de unas líneas debajo de otras empleando tabuladores.



# Variables

- Es un nombre que se asocia con una porción de la memoria del ordenador, en la que se guarda el valor asignado.
- Todas las variables han de declararse antes de usarlas, la declaración consiste en una sentencia en la que figura, el tipo de dato y el *nombre* que asignamos a la variable.

OJO: *El tipo de dato de la variable no cambiará después de creada.*

- Una vez declarada se le podrá asignar valores.

```
int x=0;  
String nombre="Angel";  
double a=3.5;  
boolean bNuevo=true;  
int[] datos;
```





# Variables

A partir de Java 10 podremos utilizar la palabra reservada var para crear objetos sin tener que definir el tipo:

```
var list = List.of(1, 2, 3);
var example = "example";
var team = new Team();
```

Java va a inferir el tipo de dato que será la variable a partir del valor que sea asignada al momento de la creación.

Si no asigna un valor, no se podrá inferir el tipo, por lo tanto no se podrá usar var.





# Tipos de datos

TIPO	POR DEFECTO	TAMAÑO	VALORES EXTREMOS
boolean	false	1 bit	true, false
byte	0	8 bits	-128 a 127
char	\u0000	16 bits	'\u0000' a '\uffff'
short	0	16 bits	-32.768 a 32.767
int	0	32 bits	-2.147.486.648 a 2.147.486.647
long	0L	64 bits	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807
float	0.0F	32 bits	-3.402823e38 a 3.402823e38
double	0.0	64 bits	-1.79769313486232e308 a 1.79769313486232e308





# Los caracteres (char)

- En Java los caracteres no están restringidos a los ASCII sino que son Unicode. Unicode es una codificación de caracteres de 16 bits que soporta la mayoría de los idiomas del mundo.
- Un carácter está siempre rodeado de comillas simples como 'A', '9', 'ñ', etc.
- Un tipo especial de carácter es la secuencia de escape, que se utilizan para representar caracteres de control o caracteres que no se imprimen.
- Para indicar un carácter Unicode que no puede ser representado en ASCII, como "ö," hemos utilizado la secuencia de escape '\u00f6'. Cada "d" en la secuencia de escape es un dígito hexadecimal.

'\u00f6' = 'ö'

Secuencia de escape	Descripción
\t	Tabulador
\b	Retroceso (backspace)
\n	Nueva línea
\r	Retorno de carro
\f	Salto de línea
\'	Comilla simple
\\"	Comilla doble
\	Barra invertida



# La clase String

- Desde el punto de vista de la programación cotidiana, uno de los tipos de datos más importantes de Java es **String**.
- String define y admite cadenas de caracteres.
- En algunos otros lenguajes de programación, una cadena o string es una matriz o array de caracteres. Este no es el caso con Java. Los Strings son objetos.

“Hola Mundo”





# Identificadores

Un identificador es un nombre que identifica a una variable, a un método o función miembro, a una clase. Todos los lenguajes tienen ciertas reglas para componer los identificadores:

- Todos los identificadores han de comenzar con una letra, el carácter subrayado ( \_ ) o el carácter dollar ( \$ ).
- Puede incluir, pero no comenzar por un número.
- No puede incluir el carácter espacio en blanco.
- Distingue entre letras mayúsculas y minúsculas.
- No se pueden utilizar las palabras reservadas como identificadores.





# Identificadores válidos / inválidos

`MyVariable`

`MYVARIABLE`

`myvariable`

`x`

`i`

`x1`

`i1`

`_myvariable`

`$myvariable`

`sum_of_array`

`javadesdecero`

`4num` // Identificador no válido porque comienza por un dígito  
`z#` // No válido porque contiene el carácter especial #  
`"Edad"` // No válido porque no puede contener comillas  
`Tom's` // No válido porque contiene el carácter '  
`año-nacimiento` // no válido porque contiene el carácter -  
`public` // no válido porque es una palabra reservada del lenguaje  
`__precio:final` // no válido porque contiene el carácter :



# Palabras reservadas

Las palabras reservadas se pueden clasificar en las siguientes categorías:

- Tipos de datos: boolean, float, double, int, char
- Sentencias condicionales: if, else, switch
- Sentencias iterativas: for, do, while, continue
- Tratamiento de las excepciones: try, catch, finally, throw
- Estructura de datos: class, interface, implements, extends
- Modificadores y control de acceso: public, private, protected, transient
- Otras: super, null, this.





# Mostrando datos en pantalla

La salida de datos por pantalla en Java es vital para comunicarse con el usuario.

Usemos `System.out.print()` y `System.out.println()` para mostrar mensajes y variables.

Nota: `System` es una clase y siempre debe ir con la "S" mayúscula.

```
System.out.println("Hola usuario bienvenido");
```





# Hola Mundo!

```
public class HolaMundo {  
  
    public static void main(String[] args) {  
  
        System.out.println("Hola Mundo!");  
  
    }  
  
}
```





# Para tener en cuenta en Java

1. El nombre del archivo .java debe ser igual al nombre de la estructura class que se encuentra dentro de ella.
2. El archivo .java sólo contiene el código fuente. Para generar el archivo que puede ejecutar la máquina virtual, es necesario pasarlo por el compilador *javac* que genera el archivo .class
3. Para ejecutar la aplicación, usamos el comando *java <Clase>* el cual se encarga de iniciar la máquina virtual y ejecutar el archivo .class con el nombre <Clase>.

## HolaMundo.java

```
public class HolaMundo {  
    public static void main(String[] args) {  
        System.out.println("Hola Mundo!");  
    }  
}
```

### Consola:

```
> javac HolaMundo.java (genera HolaMundo.class)  
> java HolaMundo (Sin el .class)
```





# Recibiendo datos del usuario

La entrada de datos o lectura de datos por teclado en Java es vital para la interacción con el usuario usemos las clases **Scanner** y **BufferedReader**.

```
Scanner sc = new Scanner(System.in);
System.out.println("Por favor ingrese su nombre");
String nombre = sc.nextLine();
```

```
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
System.out.println("Por favor ingrese su nombre");
String nombre = br.readLine();
```



# Hola, Quién?

```
public class HolaQuien {  
  
    public static void main(String[] args) {  
  
        var sc = new Scanner(System.in);  
  
        System.out.println("Por favor ingrese su nombre");  
  
        var nombre = sc.nextLine();  
  
        System.out.println("Hola " + nombre + "!");  
    }  
}
```



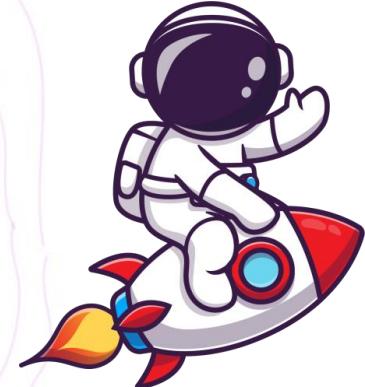
Hechos  
QUE CONECTAN ✓



# Bloques de código

- Un bloque de código es un grupo de sentencias que se comportan como una unidad.
- Un bloque de código está limitado por las llaves de apertura { y cierre }.

```
{  
    saludo="Hola mundo";  
    System.out.println(saludo);  
}
```





# Estructuras de Control

```
if (condicion) {  
    instrucciones  
} else if (condicion) {  
    instrucciones  
} else {  
    instrucciones  
}
```

```
switch (expresion) {  
    case valor1:  
        instrucciones  
        break;  
    case valor2:  
        instrucciones  
        break;  
    default:  
        instrucciones  
}
```





# Estructuras de Repetición (ciclos)

```
while (condicion) {  
    instrucciones  
}  
  
do {  
    instrucciones  
} while (condicion);  
  
for (inicializacion; condicion; incremento) {  
    instrucciones  
}
```



Sentencia **break**: es utilizada en la mayoría de los casos para interrumpir una ejecución de una estructura **switch** aunque también se puede utilizar para estructuras de repetición.

Sentencia **continue**: se utiliza para estructuras de repetición. Cuando se ejecuta, inmediatamente el resto de sentencias no se ejecutan y vuelve al comienzo de esta.



# Expresiones

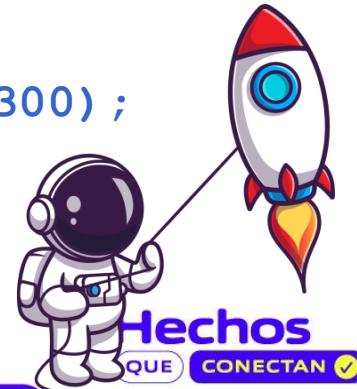
- Una expresión es todo aquello que se puede poner a la derecha del operador asignación =
- Por ejemplo:

```
x = 123;
```

```
y = (x+100)/4;
```

```
area = circulo.calcularArea(2.5);
```

```
Rectangulo r = new Rectangulo(10, 10, 200, 300);
```



Hechos

QUE CONECTAN ✓



# Operadores

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

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

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

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



# Incremento y Decremento

```
int x = 10;
System.out.println(x);      10
x++;
System.out.println(x);      11
System.out.println(++x);    12
System.out.println(x++);    12
System.out.println(x);      13
System.out.println(x++);    13
System.out.println(++x);    15
System.out.println(++x);    16
++x;
x++;
System.out.println(++x);    19
System.out.println(x++);    19
System.out.println(++x);    21
```

```
char A = 'c';
char B;
System.out.println(A++);    c
System.out.println(A++);    d
System.out.println(++A);    f
B = --A;
System.out.println(++A);    f
A++;
--B;
System.out.println(B++);    d
System.out.println(++B);    f
System.out.println(++A);    h
System.out.println(B--);    f
System.out.println(A);      h
System.out.println(B);      e
```



# Operadores

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

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
<code>?:</code>	operador condicional	<code>a = 4; b = a == 4 ? a+5 : 6-a; b = a &gt; 4 ? a*7 : a+8;</code>	b vale 9 b vale 12



# Operadores

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



# Precedencia de operadores

Prior.	Operador	Tipo de operador	Operación
1	<code>++</code> <code>--</code> <code>+, -</code> <code>~</code> <code>i</code>	Aritmético Aritmético Aritmético Integral Booleano	Incremento previo o posterior (unario) Incremento previo o posterior (unario) Suma unaria, Resta unaria Cambio de bits (unario) Negación (unario)
2	<code>(tipo)</code>	Cualquiera	
3	<code>*, /, %</code>	Aritmético	Multiplicación, división, resto
4	<code>+, -</code> <code>+</code>	Aritmético Cadena	Suma, resta Concatenación de cadenas
5	<code>&lt;&lt;</code> <code>&gt;&gt;</code> <code>&gt;&gt;&gt;</code>	Integral Integral Integral	Desplazamiento de bits a izquierda Desplazamiento de bits a derecha con inclusión de signo Desplazamiento de bits a derecha con inclusión de cero
6	<code>&lt;, &lt;=</code> <code>&gt;, &gt;=</code> <code>instanceof</code>	Aritmético Aritmético Objeto, tipo	Menor que, Menor o igual que Mayor que, Mayor o igual que Comparación de tipos
7	<code>==</code> <code>i=</code> <code>==</code> <code>i=</code>	Primitivo Primitivo Objeto Objeto	Igual (valores idénticos) Desigual (valores diferentes) Igual (referencia al mismo objeto) Desigual (referencia a distintos objetos)



# Precedencia de operadores

Prior.	Operador	Tipo de operador	Operación
8	& &	Integral Booleano	Cambio de bits AND Producto booleano
9	^ ^	Integral Booleano	Cambio de bits XOR Suma exclusiva booleana
10	 	Integral Booleano	Cambio de bits OR Suma booleana
11	&&	Booleano	AND condicional
12		Booleano	OR condicional
13	? :	Booleano, cualquiera, cualquiera	Operador condicional (ternario)
14	= *=, /=, %= +=, -= <<=, >>= >>= &=, ^=,  =	Variable, cualquiera	Asignación Asignación con operación



El futuro digital  
es de todos

MinTIC

UTP  
Universidad Tecnológica  
de Pereira

Misión  
TIC 2022

# Ejercicios para ejercitarnos



Hechos  
QUE CONECTAN ✓



# Vamos al código

- Crear un proyecto Maven para los ejercicios de la clase 1
  - Ctrl + Shift + P
  - > Java: Create Java Project...
  - No build tools
  - Java project name: clase01
- Por cada ejercicio propuesto se creará una clase, con una función estática (método) que resuelva dicho ejercicio.
- Usar la función `main()` sólo para hacer el llamado a las funciones que se quieren probar.





# Ejercicio 1

Implemente un algoritmo que dado un nombre en una variable de tipo cadena, imprima un saludo en consola.

```
-----< co.edu.utp.isc.progiv:p4-clase2 >-----
Building p4-clase2 1.0-SNAPSHOT
----- [ jar ] -----
--- exec-maven-plugin:3.0.0:exec (default-cli) @ p4-clase2 ---
Hola Cesar Díaz!
-----
BUILD SUCCESS
-----
```

```
public class Ejercicio1 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        var nombre = "Cesar Díaz";

        var resultado = saludo(nombre);

        System.out.println(resultado);
    }

    public static String saludo(String nombre) {
        return "Hola " + nombre + "!";
    }
}
```



# Ejercicio 2

Implemente un algoritmo que reciba un número por teclado y cuente cuántas cifras (o dígitos) contiene e imprima el mensaje en consola.

Entrada	Salida
15	El número tiene 2 cifras
154326	El número tiene 6 cifras
0	El número tiene 0 cifras

```
--< co.edu.utp.isc.progiv:p4-clase2 >
Building p4-clase2 1.0-SNAPSHOT
-----[ jar ]-----

--- exec-maven-plugin:3.0.0:exec (default-cli) @ p4-clase2 ---
Introduce un número entero:
0
El número tiene 0 cifras
-
BUILD SUCCESS
-
```

```
public class Ejercicio2 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        var sc = new Scanner(System.in);

        System.out.println("Introduce un número entero: ");
        var numero = sc.nextInt();

        var digitos = numeroDigitos(numero);
        System.out.println("El número tiene " + digitos + " cifras");
    }

    public static int numeroDigitos(int numero) {
        var cifras = 0;

        while (numero != 0) {
            numero /= 10;
            cifras++;
        }

        return cifras;
    }
}
```



# Ejercicios adicionales

3. Escribe un programa Java que lee un número entero por teclado y obtiene y muestra por pantalla el doble y el triple de ese número.
4. Escribe un programa que lea una cantidad de grados centígrados y la pase a grados Fahrenheit. La fórmula es:  $F = 32 + (9 * C / 5)$
5. Escribe un programa java que lea una variable de tipo entero y asígnale un valor. A continuación muestra un mensaje indicando si la variable es par o impar. Utiliza el operador condicional `( ) ? ( )` para resolverlo.  
Ej: "14 es par" o "15 es impar"





# Para la próxima sesión...

- Terminar los ejercicios que no se terminaron... (si aplica)
- Revisar el material
  - Instalación Git
  - Integración Git Github y VSCode
  - Ver videos:
    - Maven (Video 1 a 5)
    - Curso Git & GitHub

