

Estructura de datos y algoritmos

Rodrigo Alvarez

rodrigo.alvarez2@mail_udp.cl

Que es java?

Que es java?



Que es java?

- Desarrollado por Sun Microsystems
- Lanzado en 1995
- Actualmente propiedad de Oracle
- Open Source



Que es java?

- Lenguaje compilado a bytecode
(corre encima de la jvm)
- Fuertemente orientado a objetos
- Sintaxis similar a C++
- Multiplataforma



Domina todos lenguajes de programación estudiando en EDteam

ed.team/cursos



MAJOR COMPANIES THAT USE



Hello world

The screenshot shows a Java code editor interface. At the top, there's a header bar with tabs for "Main.java" and a "+" button, followed by the title "Java Hello World". On the right of the header are buttons for "NEW", "JAVA ▾", "RUN ►", and a three-dot menu. The main area has a dark background. On the left, a vertical sidebar shows line numbers starting from 1. The main code area is empty, indicated by a large gray placeholder text box. To the right of the code area, there are two sections: "STDIN" with a placeholder "Input for the program (Optional)" and "Output:" with a placeholder "Click on RUN button to see the output".

Tipo dato	Tamaño	Descripción
byte	1 byte	números del -128 al 127
short	2 bytes	-32,768 al 32,767
int	4 bytes	-2,147,483,648 al 2,147,483,647
long	8 bytes	-9,223,372,036,854,775,808 al 9,223,372,036,854,775,807
float	4 bytes	números racionales, hasta 7 dígitos decimales
double	8 bytes	hasta 15 dígitos decimales
boolean	1 bit	true o false
char	2 bytes	un único carácter

Operadores aritméticos

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

Operadores de asignación

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3

Operadores de comparación

Operator	Name	Example
<code>==</code>	Equal to	<code>x == y</code>
<code>!=</code>	Not equal	<code>x != y</code>
<code>></code>	Greater than	<code>x > y</code>
<code><</code>	Less than	<code>x < y</code>
<code>>=</code>	Greater than or equal to	<code>x >= y</code>
<code><=</code>	Less than or equal to	<code>x <= y</code>

Operadores lógicos

Operator	Name	Description	Example
&&	Logical and	true if both statements are true	<code>x < 5 && x < 10</code>
	Logical or	true if one of the statements is true	<code>x < 5 x < 4</code>
!	Logical not	Reverse the result	<code>!(x < 5 && x < 10)</code>

I/O: Input/Output en java

The screenshot shows a Java code editor interface. The top bar includes tabs for "lo.java" and "+", a file identifier "427p45t7h", and buttons for "NEW", "JAVA ▾", "RUN ▶", and a three-dot menu. The code area contains a single line of code: "1". To the right, under "STDIN", the user has entered "23". Below a horizontal line, the "Output:" section displays the program's response:

```
Output:  
Ingresa dos ints:  
el primer numero es: 2  
el segundo numero es: 3  
La suma de los numeros es: 5
```

Scanner class

La clase `Scanner` se utiliza para obtener la entrada del usuario, y es parte del paquete `java.util`.

Method	Description
<code>nextBoolean()</code>	Reads a <code>boolean</code> value from the user
<code>nextByte()</code>	Reads a <code>byte</code> value from the user
<code>nextDouble()</code>	Reads a <code>double</code> value from the user
<code>nextFloat()</code>	Reads a <code>float</code> value from the user
<code>nextInt()</code>	Reads a <code>int</code> value from the user
<code>nextLine()</code>	Reads a <code>String</code> value from the user
<code>nextLong()</code>	Reads a <code>long</code> value from the user

Scanner

The screenshot shows a Java code editor interface with the following details:

- Title Bar:** The title bar displays "ScannerEx.java" and "427nv54rs".
- Toolbar:** The toolbar includes buttons for "NEW", "JAVA ▾", "RUN ▶", and a three-dot menu.
- Code Area:** The code area contains a single line of code: "1".
- stdin Area:** The "STDIN" section shows the input provided to the program: "hola", "3", and "3.14".
- Output Area:** The "Output" section shows the program's response to the inputs:
 - "Ingrese una palabra"
 - "La palabra ingresada es: hola"
 - "Ingrese un numero"
 - "El numero ingresado es: 3"
 - "Ingrese un numero flotante"
 - "El numero flotante ingresado es: 3.14"

if, else, while, for

The screenshot shows a Java code editor interface. On the left, there is a file tab labeled "IfElseWhileForExample.java". To its right is a plus sign icon. Further right is a code identifier "427nvkqmd". At the top right, there are buttons for "NEW", "JAVA ▾", "RUN ▶", and a three-dot menu icon.

The code editor has a dark theme. The main area contains the following Java code:

```
1
```

The output window on the right shows the following interaction:

STDIN
17

Output:
the number is odd
numeros pares:
2
4
6
8
10
12
14
16
divisibles por 3:
3

Strings

The screenshot shows a Java code editor interface. The top bar includes tabs for "StringExample.java" and "+", a file identifier "427nu2xbm", and buttons for "NEW", "JAVA ▾", "RUN ▶", and a menu icon. The main area displays a Java code snippet:

```
1 public class StringExample {  
2     public static void main(String[] args) {  
3         String str = "Hello World. Goodbye";  
4         System.out.println(str);  
5     }  
6 }
```

The output window shows the standard input "23" followed by the output "Hello World. Goodbye".

STDIN
23

Output:
H
e
l
l
o

W
o
r
l
d
Hello World. Goodbye

String methods

Arrays

The screenshot shows a Java code editor interface. At the top, there are tabs for "ArrayExample.java" and a plus sign, with the identifier "427p4c9xn" next to it. On the right side, there are buttons for "NEW", "JAVA ▾", "RUN ▶", and three vertical dots. The main area has a dark background. On the left, a vertical bar has the number "1" at its top. The central area contains the following code:

```
public class ArrayExample {  
    public static void main(String[] args) {  
        int[] arr = {1, 2, 3, 4};  
        System.out.println(arr[3]);  
    }  
}
```

To the right of the code, under the heading "STDIN", is the placeholder text "Input for the program (Optional)". Below the code, under the heading "Output:", is the number "4".

Métodos

```
[modificador de acceso] [tipo_retorno] nombre_funcion(tipo_1 parametro_1, tipo_2 parametro_2, ...){  
    // instrucciones  
    return valor;  
}
```

Métodos

Main.java Greeter.java + 427p5fbvj NEW JAVA ▾ RUN ► :

1

STDIN

Input for the program (Optional)

Output:

hi!

Clases

The screenshot shows a Java code editor interface. At the top, there's a header with "ClassExample.java" on the left, a "+" button, the user ID "427pmavjh" in the center, and "NEW", "JAVA ▾", "RUN ▶", and a three-dot menu icon on the right. The main area has a dark background. On the left, a vertical toolbar has the number "1". The main code area is empty. To the right, there's a "STDIN" section with placeholder text "Input for the program (Optional)" and a green horizontal line. Below it is an "Output:" section containing the text "miauuuu".

Herencia

The screenshot shows a Java code editor interface. On the left, there is a dark sidebar with a vertical scroll bar. In the main area, the file `Car.java` is open, indicated by the tab at the top left. A plus sign (+) is next to the tab, suggesting it can be used to add another file. To the right of the tabs, the identifier `427pqrv5m` is displayed. At the top right, there are several buttons: `NEW`, a dropdown menu set to `JAVA`, a pink `RUN ▶` button, and a three-dot menu icon.

The code editor's interface is dark-themed. The code area has a light gray background with syntax highlighting. The first line of code is `1`. The code itself is not visible in the screenshot.

On the right side of the interface, there are two sections: `STDIN` and `Output`. The `STDIN` section contains the placeholder text `Input for the program (Optional)`. The `Output` section displays the results of the program execution:

```
Tuut, tuut!
Ford 2023 4
```

Interfaces

Las interfaces en Java son una colección de métodos abstractos y constantes que pueden ser implementados por cualquier clase que las implemente.

Una interfaz define un conjunto de métodos y su firma, pero no proporciona implementaciones de los métodos.

Interfaces

The screenshot shows a Java code editor interface. On the left, there's a dark sidebar with a file icon and a number '1'. The main area has tabs for 'Main.java' and '+'. The title bar shows '427pruj44'. On the right, there are buttons for 'NEW', 'JAVA ▾', 'RUN ▶', and three dots. The code editor area is mostly empty. To the right of the editor is a panel with 'STDIN' and a placeholder 'Input for the program (Optional)'. Below this is a horizontal line. Underneath the line is the word 'Output:' followed by the text 'The pig says: wee wee Zzz'.

```
Main.java
```

```
+ 427pruj44
```

```
STDIN
```

```
Input for the program ( Optional )
```

```
Output:
```

```
The pig says: wee wee  
Zzz
```

Generics

The screenshot shows a Java code editor interface. On the left, there is a file tab labeled "Main.java". To its right is a "+" button. In the center, the code editor displays the following Java code:

```
427psdcfw
1
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

At the top right of the editor, there are several buttons: "NEW", "JAVA ▾", "RUN ►", and a three-dot menu icon.

On the right side of the interface, there are two sections: "STDIN" and "Output".

The "STDIN" section contains the placeholder text "Input for the program (Optional)".

The "Output" section contains the text "Output:" followed by the number "3".

- `ArrayList`: Implementa un **TDA List** en array manipulando el tamaño de forma dinámica.
- `LinkedList`: Implementa un **TDA List** sobre una lista de nodos.
- `Queue`: Implementa un **TDA FIFO** en una lista de nodos.
- `Stack`: Implementa un **TDA LIFO** en una lista de nodos.
- `Map`: Implementa una estructura **TDA key-value** en un árbol o una tabla hash.
- `Set`: Implementa una estructura **TDA unique value** en un árbol o una tabla hash.

