



Guía rápida de fundamentos en **Java**

Con más de 200 términos

mouredev^{pro}

mouredev.pro/recursos

Índice

Introducción	2
Tipos de datos básicos	3
Sintaxis básica (parte 1 de 2)	4
Sintaxis básica (parte 2 de 2)	5
Operadores (parte 1 de 2)	6
Operadores (parte 2 de 2)	7
Métodos/Funciones (parte 1 de 2)	8
Métodos/Funciones (parte 2 de 2)	9
Estructuras de datos (parte 1 de 2)	10
Estructuras de datos (parte 2 de 2)	11
Manejo de archivos	12
Paquetes estándar	13
Librerías y Frameworks externos	14
¿Quieres aprender más sobre Java?	15

Introducción

Esta es una guía de referencia que hace un recorrido por los principales fundamentos del lenguaje de programación Java.

Puedes utilizarla para conocer rápidamente las características de Java, apoyar tu aprendizaje y obtener información sobre más de 200 términos de manera rápida y sencilla, consultando nombres, definiciones y sintaxis.

Tipos de datos básicos

Los **tipos de datos** son la base de cualquier programa, ya que determinan cómo se almacenan y manipulan los valores. Java es un lenguaje fuertemente tipado que distingue entre tipos primitivos (datos simples) y tipos de referencia (objetos complejos), como *Strings*, *Arrays* o las propias *clases* que creamos. Conocerlos es fundamental para escribir código robusto y eficiente. Cada instrucción escrita en Java finaliza con punto y coma (;).

Nombre	Palabra reservada	Sintaxis
Primitivos		
Entero	int (32 bits)	0
Entero (byte)	byte (8 bits)	0
Entero (short)	short (16 bits)	0
Entero (long)	long (64 bits)	0L
Flotante	float (32 bits)	3.14
Flotante (double)	double (64 bits)	3.14
Carácter	char	'A'
Booleano	boolean	true/false
Objetos		
Cadena de texto	String	"Hola"
Array	tipo[]	{1, 2 ,3}
Lista	List	new ArrayList<>()
Conjunto	Set	new HashSet<>()
Mapa	Map	new HashMap<>()
Clase (Objeto)	NombreClase	new Object()
Nulo	null (ausencia de objeto)	null

Sintaxis básica (parte 1 de 2)

La **sintaxis** de Java es estructurada y **orientada a objetos**, heredada en gran parte de C++. Aquí encontrarás las estructuras fundamentales del lenguaje, como la definición de clases, condicionales, bucles y métodos, junto con su respectiva sintaxis para escribir código claro y organizado.

Nombre	Palabra reservada	Sintaxis
Variable	tipo / var	tipo variable = valor var variable = valor
Constante	final	final tipo constante = valor
Condicional if	if	if (condición) { }
Condicional else if	else if	else if (otra_condición) { }
Condicional else	else	else { }
Bucle for	for	for (inicialización; condición; incremento) { }
Bucle for each	for	for (elemento : colección) { }
Bucle while	while	while (condición) { }
Bucle do while	do while	do { } while (condición)
Switch	switch	switch (variable) {case valor: }
Función (método)	void / tipo	public void/tipoRetorno nombreMétodo (parámetros) { }
Método principal	main	public static void main (String[] args) { }
Clase	class	public class NombreClase { }
Sentencia break	break	break
Sentencia continue	continue	continue
Retorno	return	return valor

Sintaxis básica (parte 2 de 2)

Nombre	Palabra reservada	Sintaxis
Paquete	package	package com.paquete
Importar	import	import java.util.List
Constructor	NombreClase	public NombreClase() { }
Instanciar objeto	new	new MiClase()
Excepciones	try, catch	try (...) catch (Excepción) { }
Bloque finally	finally	finally { }
Lanzar excepción	throw	throw new Excepción
Declarar excepción	throws	throws Excepción { }
Comprobar instancia	instanceof	variable instanceof tipo
Acceder a superclase	super	super.metodoSuperclase()
Referencia a instancia	this	this.miVariable
Modificadores de acceso		
public	public	Accesible desde cualquier otra clase en cualquier paquete.
private	private	Accesible sólo desde la clase en la que fue declarado.
protected	protected	Accesible desde el mismo paquete y subclases.
default	(sin palabra)	Por defecto. Accesible para las clases del mismo paquete.

Operadores (parte 1 de 2)

Los **operadores** son símbolos que permiten realizar cálculos y comparaciones en Java. Se dividen en varias categorías: aritméticos (suma, resta), de comparación (mayor, menor, igual), lógicos (&&, ||, !), de asignación (=, +=), bitwise y el operador ternario.

Nombre	Representación	Sintaxis
Aritméticos		
Suma	+	a + b
Resta	-	a - b
Multiplicación	*	a * b
División	/	a / b
Módulo (residuo)	%	a % b
Incremento	++	a++ / ++a
Decremento	--	a-- / --a
Comparación		
Igual a	==	a == b
Distinto de	!=	a != b
Mayor que	>	a > b
Menor que	<	a < b
Mayor o igual que	>=	a >= b
Menor o igual que	<=	a <= b

Operadores (parte 2 de 2)

Nombre	Representación	Sintaxis
Lógicos		
AND	&&	a && b
OR		a b
NOT	!	!a
Asignación		
Asignación	=	x = 5
A. con suma	+=	x += 3
A. con resta	-=	x -= 3
A. con multiplicación	*=	x *= 3
A. con división	/=	x /= 3
Bitwise		
AND bit a bit	&	a & b
OR bit a bit		a b
XOR bit a bit	^	a ^ b
Desplaz. a la izq.	<<	a << n
Desplaz. a la der.	>>	a >> n
Ternario		
Condicional ternario	? :	condición ? valorSiTrue : valorSiFalse

Métodos/Funciones (parte 1 de 2)

En Java, la funcionalidad está encapsulada en **métodos (funciones)** dentro de clases. La librería estándar proporciona clases con métodos muy útiles para tareas comunes, desde la manipulación de texto con *String* hasta operaciones matemáticas con *Math* o la interacción con la consola a través de *System*.

Nombre	Clase	Operación
<code>println()</code>	<code>System.out</code>	Muestra texto o variables en la consola con un salto de línea.
<code>equals()</code>	<code>Object</code>	Compara si un objeto es igual a otro (contenido).
<code>toString()</code>	<code>Object</code>	Devuelve una representación en texto del objeto.
<code>hashCode()</code>	<code>Object</code>	Devuelve un código hash del objeto.
<code>length()</code>	<code>String</code>	Devuelve la cantidad de caracteres de una cadena.
<code>charAt()</code>	<code>String</code>	Devuelve el carácter en una posición específica.
<code>substring()</code>	<code>String</code>	Extrae una parte de una cadena.
<code>toUpperCase()</code>	<code>String</code>	Convierte una cadena a mayúsculas.
<code>toLowerCase()</code>	<code>String</code>	Convierte una cadena a minúsculas.
<code>trim()</code>	<code>String</code>	Elimina los espacios en blanco al inicio y al final.
<code>split()</code>	<code>String</code>	Divide una cadena en un array de subcadenas.
<code>contains()</code>	<code>String</code>	Comprueba si una cadena contiene una subcadena.
<code>valueOf()</code>	<code>String</code>	Convierte un tipo primitivo a su representación en <code>String</code> .

Métodos/Funciones (parte 2 de 2)

Nombre	Clase	Operación
<code>abs()</code>	Math	Retorna el valor absoluto de un número.
<code>sqrt()</code>	Math	Calcula la raíz cuadrada.
<code>pow()</code>	Math	Calcula una potencia.
<code>random()</code>	Math	Genera un número aleatorio entre 0.0 y 1.0.
<code>max()</code>	Math	Devuelve el valor máximo entre dos números.
<code>min()</code>	Math	Devuelve el valor mínimo entre dos números.
<code>round()</code>	Math	Redondea al entero más cercano.
<code>parseInt()</code>	Integer	Convierte una cadena a un entero (int).
<code>parseDouble()</code>	Double	Convierte una cadena a un double
<code>now()</code>	LocalDate / LocalDateTime	Obtiene la fecha o fecha y hora actual.
<code>of()</code>	LocalDate / LocalTime	Crea una fecha/hora a partir de valores.
<code>format()</code>	DateTimeFormatter	Formatea una fecha/hora a una cadena.

Estructuras de datos (parte 1 de 2)

El Java Collections Framework (JCF) provee un conjunto robusto de **estructuras de datos**. Las Listas, Conjuntos y Mapas son las interfaces principales, con implementaciones como *ArrayList*, *HashSet* y *HashMap* que ofrecen distintas garantías de rendimiento y orden. También existe el *Array* como estructura primitiva.

Nombre	Operación	Sintaxis
Arrays		
length	Obtiene el tamaño (propiedad)	array.length
Listas (ArrayList)		
add()	Agrega múltiples elementos	lista.add(valor)
add()	Inserta en una posición	lista.add(índice, valor)
get()	Obtiene un elemento por índice	lista.get(índice)
set()	Modifica un elemento por índice	lista.set(índice, nuevo)
remove()	Elimina un elemento por índice	lista.remove(índice)
remove()	Elimina la primera ocurrencia	lista.remove(valor)
size()	Devuelve el tamaño	lista.size()
indexOf()	Devuelve el índice de un valor	lista.indexOf(valor)
clear()	Elimina todos los elementos	lista.clear()
contains()	Comprueba si existe el valor	lista.contains(valor)
Colecciones		
sort()	Ordena un array o una lista	lista.sort()
filter()	Filtra elementos según condición	lista.filter()
map()	Transforma cada elemento	lista.map()

Estructuras de datos (parte 2 de 2)

Nombre	Operación	Sintaxis
Conjuntos (HashSet)		
add()	Agrega un elemento (si no existe)	set.add(valor)
remove()	Elimina un elemento	set.remove(valor)
contains()	Comprueba si existe el valor	set.contains(valor)
size()	Devuelve el número de elementos	set.size()
isEmpty()	Comprueba si está vacío	set.isEmpty()
clear()	Elimina todos los elementos	set.clear()
Mapas (HashMap)		
put()	Inserta/actualiza un par clave-valor	mapa.put(clave, valor)
get()	Obtiene el valor de una clave	mapa.get(clave)
remove()	Elimina el par de una clave	mapa.remove(clave)
containsKey()	Comprueba si existe la clave	mapa.containsKey(clave)
containsValue()	Comprueba si existe el valor	mapa.containsValue(val)
keySet()	Devuelve un Set de las claves	mapa.keySet()
values()	Devuelve una Collection de valores	mapa.values()
entrySet()	Devuelve un Set de pares clave-valor	mapa.entrySet()
size()	Devuelve el número de pares	mapa.size()
clear()	Elimina todos los pares	mapa.clear()

Manejo de archivos

Java ofrece un potente sistema de **manejo de archivos** a través de los paquetes *java.io* (clásico, basado en streams) y *java.nio* (moderno, más eficiente). Es recomendable usar el bloque *try-with-resources* para asegurar que los ficheros se cierren automáticamente.

Nombre	Representación	Sintaxis
Moderno (java.nio)		
<code>readString()</code>	Lee todo el contenido a un String	<code>Files.readString(path)</code>
<code>readAllLines()</code>	Lee todas las líneas a una List	<code>Files.readAllLines(path)</code>
<code>writeString()</code>	Escribe un String en un archivo	<code>Files.writeString(path)</code>
<code>exists()</code>	Comprueba si un archivo existe	<code>Files.exists(path)</code>
<code>createDirectory()</code>	Crea un directorio	<code>Files.createDirectory(p)</code>
<code>delete()</code>	Elimina	<code>Files.delete(path)</code>
Clásico (java.io)		
<code>FileReader</code>	Lee un archivo de texto	<code>new FileReader(file)</code>
<code>BufferedReader</code>	Lee texto de forma eficiente	<code>new BufferedReader(reader)</code>
<code>readLine()</code>	Lee una línea	<code>bufferedReader.readLine()</code>
<code>FileWriter</code>	Escribe en un archivo	<code>new FileWriter(file)</code>
<code>BufferedWriter</code>	Escribe texto de forma eficiente	<code>new BufferedWriter(writer)</code>
<code>write()</code>	Escribe una cadena	<code>bufferedWriter.write(text)</code>

Paquetes estándar

Java cuenta con una **Biblioteca de Clases Estándar** (API) muy extensa, organizada en **paquetes** (*packages*). Estos paquetes proveen funcionalidades listas para usar sin necesidad de instalar librerías externas, desde colecciones y concurrencia hasta redes y acceso a bases de datos. Aquí tienes algunos.

Nombre	Descripción
<code>java.lang</code>	Paquete fundamental, importado automáticamente. Contiene Object, String, System, Math, y las clases envoltorio (Integer, Double).
<code>java.util</code>	Contiene el Collections Framework (List, Map, Set), clases de utilidad como Scanner, Random, Optional y Objects.
<code>java.nio</code>	Nueva API de I/O (NIO), ofrece manejo de archivos y redes de alto rendimiento, con buffers y canales.
<code>java.time</code>	API moderna y completa para el manejo de fechas, horas, duraciones e instantes. Sustituye a la antigua <code>java.util.Date</code> .
<code>java.net</code>	Proporciona clases para la programación de red, como Socket, URL, URLConnection para crear clientes y servidores.
<code>java.math</code>	Ofrece clases para cálculos matemáticos.
<code>java.util.stream</code>	La API de Streams, que permite un procesamiento de datos de estilo funcional (declarativo) sobre colecciones.
<code>java.sql</code>	API de JDBC para interactuar con bases de datos relacionales mediante sentencias SQL.
<code>javax.swing</code>	Framework para crear interfaces gráficas de usuario (GUI) de escritorio. Ha sido en gran parte sucedido por JavaFX.
<code>javafx.*</code>	Framework moderno para crear aplicaciones de escritorio enriquecidas y multiplataforma. Es el sucesor de Swing.

Librerías y Frameworks externos

El ecosistema de Java es inmenso. Para gestionar dependencias y construir proyectos, se usan herramientas como **Maven** o **Gradle**. Estas permiten incorporar **librerías** y **frameworks** que extienden las capacidades del lenguaje para todo tipo de aplicaciones.

Nombre	Descripción
Spring	El framework más popular para crear aplicaciones empresariales robustas y escalables, especialmente con Spring Boot para microservicios y web.
Hibernate	Framework de Mapeo Objeto-Relacional (ORM) que simplifica la interacción con bases de datos SQL al trabajar con objetos Java.
JUnit	El estándar para realizar pruebas unitarias en Java.
Mockito	Librería para crear objetos "mock" (simulados) en pruebas unitarias, permitiendo aislar el código a probar.
Apache Commons	Conjunto de librerías con utilidades reusables para operaciones con cadenas, colecciones, I/O, etc.
Quarkus / Micronaut	Frameworks modernos y ligeros, optimizados para microservicios, serverless y compilación nativa con GraalVM.
Jackson / Gson	Librerías para serializar objetos Java a formato JSON y deserializar JSON a objetos Java.
SLF4J / Logback	Fachada de logging (SLF4J) y una implementación (Logback/Log4j2) para un registro de eventos flexible y potente.
Jakarta EE	Evolución de Java EE, es un conjunto de especificaciones para crear aplicaciones empresariales.
Apache Kafka	Aunque es una plataforma de mensajería y transmisión de datos, sus clientes Java son esenciales para construir sistemas de streaming de eventos en tiempo real.

¿Quieres aprender más sobre Java?

Aquí tienes mis cursos para aprender Java desde cero.

Curso gratis en YouTube:

<https://mouredev.link/java>

Curso con extras (lecciones por tema, ejercicios, soporte, comunidad, test y certificado) en el campus de estudiantes mouredev pro:

<https://mouredev.pro/cursos/java-poo-desde-cero>

(Utiliza el cupón "PRO" para acceder con un 10% de descuento a todas las suscripciones y cursos del campus).

mouredev^{pro}

Estudia programación y desarrollo de software de manera diferente

mouredev.pro