Guía Rápida de Fundamentos en Java

Instructor

I.S.C. E.T.W. M.T.W. Carlos Uriel de Jesús Sánchez González

https://www.linkedin.com/in/carlos-uriel-de-jesus-sanchez-gonzalez-8920a1372/

Tabla de contenido

Introducción 3
Tipos de Datos Básicos 3
Sintaxis básica (parte 1 de 2) 4
Sintaxis básica (parte 2 de 2) 6
Operadores (parte 1 de 2) 7
Operadores (parte 2 de 2) 8
Métodos/Funciones (parte 1 de 2) 10
Métodos/Funciones (parte 2 de 2) 11
Estructuras de datos (parte 1 de 2) 12
Estructuras de datos (parte 2 de 2) 13
Manejo de archivos 15
Paquetes estándar 16
Librerías y Frameworks externos 17

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 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
`println()`	`System.out`	Muestra texto o variables en la consola con un salto de línea.
`equals()`	`Object`	Compara si un objeto es igual a otro (contenido).
`toString()`	`Object`	Devuelve una representación en texto del objeto.
`hashCode()`	`Object`	Devuelve un código hash del objeto.
`length()`	`String`	Devuelve la cantidad de caracteres de una cadena.
`charAt()`	`String`	Devuelve el carácter en una posición específica.
`substring()`	`String`	Extrae una parte de una cadena.
`toUpperCase()`	`String`	Convierte una cadena a mayúsculas.
`toLowerCase()`	`String`	Convierte una cadena a minúsculas.
`trim()`	`String`	Elimina los espacios en blanco al inicio y al final.

`split()`	`String`	Divide una cadena en un array de subcadenas.
`contains()`	`String`	Comprueba si una cadena contiene una subcadena.
`valueOf()`	`String`	Convierte un tipo primitivo a su representación en String.

Métodos/Funciones (parte 2 de 2)

Nombre	Clase	Operación
`abs()`	`Math`	Retorna el valor absoluto de un número.
`sqrt()`	`Math`	Calcula la raíz cuadrada.
`pow()`	`Math`	Calcula una potencia.
`random()`	`Math`	Genera un número aleatorio entre 0.0 y 1.0.
`max()`	`Math`	Devuelve el valor máximo entre dos números.
`min()`	`Math`	Devuelve el valor mínimo entre dos números.
`round()`	`Math`	Redondea al entero más cercano.
`parseInt()`	`Integer`	Convierte una cadena a un entero (int).

`parseDouble()`	`Double`	Convierte una cadena a un double
`now()`	`LocalDate / LocalDateTime`	Obtiene la fecha o fecha y hora actual.
`of()`	`LocalDate / LocalTime`	Crea una fecha/hora a partir de valores.
`format()`	`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)		
`readString()`	Lee todo el contenido a un String	`Files.readString(path)`
`readAllLines()`	Lee todas las líneas a una List	`Files.readAllLines(path)`
`writeString()`	Escribe un String en un archivo	`Files.writeString(path)`
`exists()`	Comprueba si un archivo existe	`Files.exists(path)`
`createDirectory()`	Crea un directorio	`Files.createDirectory(p)`
`delete()`	Elimina	`Files.delete(path)`
Clásico (java.io)		
`FileReader`	Lee un archivo de texto	`new FileReader(file)`
`BufferedReader`	Lee texto de forma eficiente	`new BufferedReader(reader)`
`readLine()`	Lee una línea	`bufferedReader.readLine()`

`FileWriter`	Escribe en un archivo	`new FileWriter(file)`
`BufferedWriter`	Escribe texto de forma eficiente	`new BufferedWriter(writer)`
`write()`	Escribe una cadena	`bufferedWriter.write(text)`

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
`java.lang`	Paquete fundamental, importado automáticamente. Contiene Object, String, System, Math, y las clases envoltorio (Integer, Double).
`java.util`	Contiene el Collections Framework (List, Map, Set), clases de utilidad como Scanner, Random, Optional y Objects.
`java.nio`	Nueva API de I/O (NIO), ofrece manejo de archivos y redes de alto rendimiento, con buffers y canales.
`java.time`	API moderna y completa para el manejo de fechas, horas, duraciones e instantes. Sustituye a la antigua java.util.Date.
`java.net`	Proporciona clases para la programación de red, como Socket, URL, URLConnection para crear clientes y servidores.

`java.math`	Ofrece clases para cálculos matemáticos.
`java.util.stream`	La API de Streams, que permite un procesamiento de datos de estilo funcional (declarativo) sobre colecciones.
`java.sql`	API de JDBC para interactuar con bases de datos relacionales mediante sentencias SQL.
`javax.swing`	Framework para crear interfaces gráficas de usuario (GUI) de escritorio. Ha sido en gran parte sucedido por JavaFX.
`javafx.*`	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.