En una nueva sección de aprender Java desde cero conoceremos los **tipos de datos en Java** junto con algunos ejemplos prácticos.

Existen principalmente dos tipos de lenguajes.

* El primero es el **lenguaje de tipado estático** donde cada variable y tipo de expresión ya se conoce en tiempo de compilación. Una vez que se declara una variable es de un cierto tipo de datos, no puede contener valores de otros tipos de datos. Ejemplo: C, C ++, Java.
* El segundo, son l**enguajes de tipado dinámico**: estos lenguajes pueden recibir diferentes tipos de datos a lo largo del tiempo. Ejemplo: Ruby, Python.

1. Tipos de Datos

**Java está tipado estáticamente** y es fuertemente tipado porque en Java, cada tipo de datos (como entero, carácter, hexadecimal, decimal empaquetado, etc.) está predefinido como parte del lenguaje de programación y todas las constantes o variables definidas para un programa dado debe describirse con uno de los tipos de datos.

Java tiene dos categorías de datos:

* Datos Primitivos (p. Ej., int, char)
* Datos Objeto (tipos creados por el programador)

Los tipos de datos utilizados con más frecuencia en las [declaraciones de variables Java](https://javadesdecero.es/basico/declaracion-y-tipos-variables/) se enumeran en la siguiente tabla, junto con una breve descripción:

2. Datos Primitivos

Los datos primitivos son solo valores únicos; ellos no tienen capacidades especiales. Java soporta **8 tipos de datos primitivos**:

| Tabla Datos Primitivos en Java. | | | | |
| --- | --- | --- | --- | --- |
| **TIPO** | **DESCRIPCIÓN** | **DEFAULT** | **TAMAÑO** | **EJEMPLOS** |
| boolean | true o false | false | 1 bit | true, false |
| byte | entero complemento de dos | 0 | 8 bits | 100, -50 |
| char | carácter unicode | \u0000 | 16 bits | ‘a’, ‘\u0041’, ‘\101’, ‘\\’ |
| short | entero complemento de dos | 0 | 16 bits | 10000,-20000 |
| int | entero complemento de dos | 0 | 32 bits | 100000,-2,-1,0,1,2,-200000 |
| long | entero complemento de dos | 0 | 64 bits | -2L,-1L,0L,1L,2L |
| float | coma flotante IEEE 754 | 0.0 | 32 bits | 1.23e100f, -1.23e-100f, .3ef, 3.14f |
| double | coma flotante IEEE 754 | 0.0 | 64 bits | 1.2345e300d, -1.2345e-300f, 1e1d |

Los tipos de datos primitivos se pueden organizar en 4 grupos:

* **Numéricos enteros**: Son los tipos *byte*, *short*, *int* y *long*. Los 4 representan números enteros con signo.
* **Carácter:** El tipo *char* representa un carácter codificado en el sistema unicode.
* **Numérico decimal**: Los tipos *float* y *double* representan números decimales en coma flotante.
* **Lógicos**: El tipo *boolean* es el tipo de dato lógico; los dos únicos posibles valores que puede representar un dato lógico son *true* y *false*. *true* y *false* son [palabras reservadas de Java](https://www.javadesdecero.es/identificadores-y-palabras-reservadas/).

Pese a que String no es un tipo de dato primitivo, también es utilizado al declarar variables.

[×](https://javadesdecero.es/basico/tipos-datos-java-ejemplos/)**Nota**: Recuerde que los valores de datos de **char** siempre deben estar rodeados por comillas simples, y los valores de datos de **String** siempre deben estar rodeados por comillas dobles.

### **2.1. Tipo de dato boolean**

El tipo de datos booleano representa solo un bit de información: true (verdadero) o false (falso). Los valores de tipo booleano no se convierten implícita o explícitamente (con casts) en ningún otro tipo. Pero, el programador puede escribir fácilmente el código de conversión.

Por ejemplo:

// Un programa Java para demostrar el tipo de datos booleanos

class JavadesdeCero

{

public static void main(String args[])

{

boolean b = true;

if (b == true)

System.out.println("Hola Javeros!");

}

}

Salida:

Hola Javeros!

Ahora miremos este ejemplo:

//Demostrar valores booleanos

class DemoBoolean

{

public static void main(String args[])

{

boolean b;

b=false;

System.out.println("b es: "+b);

b=true;

System.out.println("b es: "+b);

//Un valor booleano puede contralar la sentencia if

if(b) System.out.println("Esto si ejecutará.");

b=false;

if(b) System.out.println("Esto no se ejecutará.");

//El resultado de un operador relacional es un valor booleano

System.out.printf("10 > 9 es "+ (10 > 9));

}

}

Salida:

b es: false

b es: true

Esto si ejecutará.

10 > 9 es true

Hay tres cosas interesantes para notar sobre este último programa.

* Primero, como puedas ver, cuando println() arroja un valor booleano, se muestra “true” o “false”.
* En segundo lugar, el valor de una variable booleana es suficiente, por sí mismo, para controlar el  
  declaración. No es necesario escribir una instrucción if como en el primer ejemplo.
* En tercer lugar, el resultado de un operador relacional, como <, es un valor booleano. Esto es  
  por qué la expresión 10 > 9 muestra el valor “true“. Además, el paréntesis alrededor de 10 > 9 es necesario porque el operador **+** tiene una mayor precedencia que la ***>***.

### **2.2. Tipo de dato byte**

El tipo de datos byte es un entero de 8 bits de complemento a dos (una forma de representar datos positivos y negativos en binario). El tipo de datos de byte es útil para guardar en memoria grandes arrays.

* **Tamaño**: 8 bits
* **Valor**: -128 a 127

Por ejemplo:

// Programa Java para demostrar el tipo de datos de bytes

class JavadesdeCero

{

public static void main(String args[])

{

byte a = 126;

// byte tiene un valor de 8 bits

System.out.println(a);

a++;

System.out.println(a);

// Se desborda aquí porque

// el byte puede contener valores de -128 a 127

a++;

System.out.println(a);

// bucle dentro del rango

a++;

System.out.println(a);

}

}

Salida

126

127

-128

-127

### **2.3. Tipo de dato short**

El tipo de datos short es un entero de complemento de dos formado por 16 bits. De forma similar al byte, use short para guardar en memoria arrays grandes, en situaciones donde el ahorro de memoria realmente importa.

* **Tamaño**: 16 bit
* **Valor**: -32,768 a 32,767 (inclusive)

### **2.4. Tipo de dato int**

Es un entero de complemento de dos formado de 32 bits.

* **Tamaño**: 32 bits
* **Valor**: (-2^31) a (2^31 – 1)

[×](https://javadesdecero.es/basico/tipos-datos-java-ejemplos/)**Nota**: En Java SE 8 y posterior, podemos usar el tipo de datos int para representar un entero de 32 bits sin signo, que tiene un valor en el rango . Utilice la clase Integer para usar el tipo de datos int como un entero sin signo.

### **2.5. Tipo de dato long**

El tipo de datos long es un entero de complemento de dos de 64 bits.

* **Tamaño**: 64 bit
* **Valor**: (-2^63) a (2^63-1)

[×](https://javadesdecero.es/basico/tipos-datos-java-ejemplos/)**Nota**: En Java SE 8 y posteriores, puede usar el tipo de datos long para representar un long sin signo de 64 bits, que tiene un valor mínimo de 0 y un valor máximo de 264 -1. La clase long también contiene métodos como compareUnsigned, divideUnsigned, etc. para admitir operaciones aritméticas.

### **2.6. Tipo de dato float y double**

El tipo de dato **float**es una coma flotante IEEE 754 de precisión simple de 32 bits. Use float (en lugar double) si necesita guardar en memoria grandes arrays de números de coma flotante.

* **Tamaño**: 32 bits
* **Sufijo**: F/f Ejemplo: 9.8f

El tipo de dato **double** es una comna flotante IEEE 754 de 64 bits de doble precisión. Para valores decimales, este tipo de datos generalmente es la opción predeterminada.

[×](https://javadesdecero.es/basico/tipos-datos-java-ejemplos/)**Nota**: Ambos tipos de datos float y double se diseñaron especialmente para cálculos científicos, donde los errores de aproximación son aceptables. Si la precisión es la preocupación más importante en ese momento, se recomienda no utilizar estos tipos de datos y, en su lugar, utilizar la clase BigDecimal.

### **2.7. Tipo de dato char**

El tipo de datos char es un simple carácter Unicode de 16 bits. Un char es un solo carácter.

* **Valor**: ‘\ u0000’ (o 0) a ‘\ uffff’ 65535

[**¿Por qué Java usa Unicode?**](https://javadesdecero.es/basico/tipos-datos-java-ejemplos/#648386966-panel)

Java fue diseñado para uso mundial. Por lo tanto, necesita utilizar un juego de caracteres que pueda representar todos los idiomas del mundo. Unicode es el conjunto de caracteres estándar diseñado expresamente para este fin.

Por ejemplo:

// Programa Java para demostrar tipos de datos primitivos

class JavadesdeCero

{

public static void main(String args[])

{

// declarando el carácter

char a = 'G';

// El tipo de datos enteros es generalmente

// utilizado para valores numéricos

int i=89;

// use byte y short si la memoria es una prioridad

byte b = 4;

// esto dará error ya que el número es

// mayor que el rango de bytes

// byte b1 = 7888888955;

short s = 56;

// esto dará error ya que el número es

// más grande que el rango de short

// short s1 = 87878787878;

// por defecto, el valor de la fracción es double en Java

double d = 4.355453532;

// para float use 'f' como sufijo

float f = 4.7333434f;

System.out.println("char: " + a);

System.out.println("integer: " + i);

System.out.println("byte: " + b);

System.out.println("short: " + s);

System.out.println("float: " + f);

System.out.println("double: " + d);

}

}

Salida:

char: G

integer: 89

byte: 4

short: 56

float: 4.7333436

double: 4.355453532