

Tipos de datos

En Java existen ocho tipos de datos, también conocidos como tipos primitivos :
byte, short, int, long, char, float, double y boolean.

Enteros

Este grupo incluye *byte*, *short*, *int* y *long*.

Estos tipos de datos nos permiten trabajar con números enteros ya sean positivos o negativos.

Nombre	bytes	Rango
long	8	−9,223,372,036,854,775,808 a 9,223,372,036,854,775,807
int	4	−2,147,483,648 a 2,147,483,647
short	2	−32,768 to 32,767
byte	1	−128 to 127

En la mayoría de los casos usaremos el tipo de dato int para números positivos.

Ejemplos

```
byte diasMes = 30;
```

```
short diasAño = (12 * 30);
```

```
int velocidadLuz = 300000000;
```

```
long añoLuz = 300000000 * 365;
```

Flotantes/Decimales

Este grupo incluye *float* y *double*. Estos tipos de datos nos permiten trabajar con números los cuales posean punto decimal ya sean positivos o negativos.

Nombre	bytes	Rango Aproximado
Double	8	4.9e−324 to 1.8e+308
float	4	1.4e−045 to 3.4e+038

Ejemplos

```
float pi = 3.1415926535f;
```

```
double e = 2.718281828459045235360;
```

Caracteres

En el grupo de caracteres únicamente encontraremos el tipo de dato char. Este tipo de datos nos permitirá trabajar con caracteres.

Ejemplos

```
char letraA = 'a';
```

```
char letraANumerico = 77;
```

Booleanos

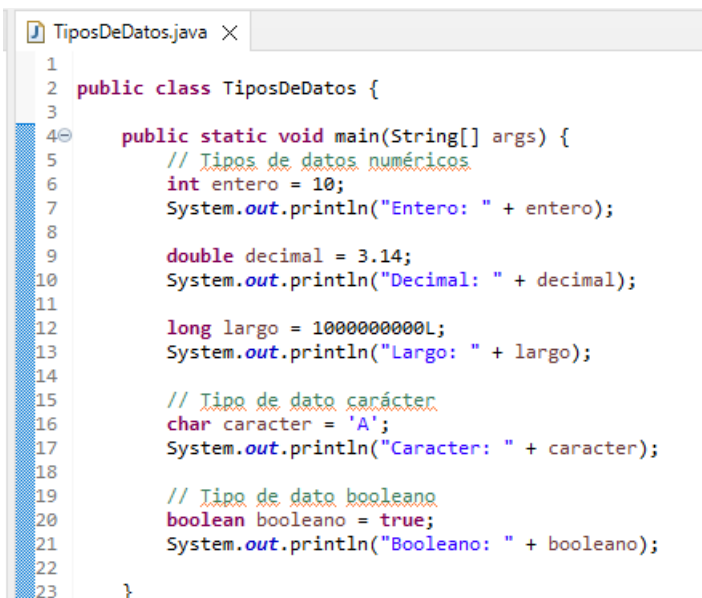
Al igual que el grupo de caracteres, en el grupo de booleanos únicamente encontraremos un tipo de dato, el boolean. Este tipo de datos nos permitirá trabajar con valores lógicos, verdadero o falso.

Ejemplos

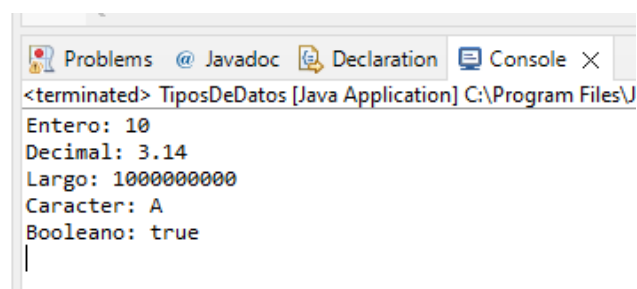
```
boolean verdadero = true;
```

```
boolean falso = false;
```

Por ejemplo podríamos mostrar por consola los datos



```
1
2 public class TiposDeDatos {
3
4     public static void main(String[] args) {
5         // Tipos de datos numéricos
6         int entero = 10;
7         System.out.println("Entero: " + entero);
8
9         double decimal = 3.14;
10        System.out.println("Decimal: " + decimal);
11
12        long largo = 1000000000L;
13        System.out.println("Largo: " + largo);
14
15        // Tipo de dato carácter
16        char caracter = 'A';
17        System.out.println("Caracter: " + caracter);
18
19        // Tipo de dato booleano
20        boolean booleano = true;
21        System.out.println("Booleano: " + booleano);
22    }
23 }
```



```
<terminated> TiposDeDatos [Java Application] C:\Program Files\J
Entero: 10
Decimal: 3.14
Largo: 1000000000
Caracter: A
Booleano: true
|
```

Variables

El uso de variables es fundamental en la programación por varias razones:

1. Almacenamiento de datos: Las variables permiten almacenar y guardar datos en la memoria durante la ejecución de un programa. Puedes guardar valores numéricos, textuales, booleanos y otros tipos de datos en variables para su posterior uso.
2. Manipulación y cálculos: Las variables permiten realizar operaciones y cálculos con los datos almacenados en ellas. Puedes realizar sumas, restas, multiplicaciones, divisiones y otras operaciones utilizando variables, lo que facilita la manipulación y transformación de los datos según las necesidades del programa.
3. Reutilización de datos: Utilizar variables te permite almacenar un valor una vez y reutilizarlo en diferentes partes de tu programa. En lugar de repetir el mismo valor en varias ocasiones, puedes almacenarlo en una variable y referenciarla en diferentes lugares, lo que hace que tu código sea más conciso y fácil de mantener.
4. Flexibilidad y adaptabilidad: Las variables permiten que tu programa sea más flexible y adaptable a diferentes situaciones. Puedes cambiar el valor de una variable en tiempo de ejecución, lo que afectará el comportamiento del programa. Esto te permite ajustar y adaptar la lógica del programa según las condiciones y requisitos cambiantes.
5. Mejora la legibilidad y comprensión del código: El uso adecuado de variables con nombres descriptivos mejora la legibilidad y comprensión del código. Al utilizar nombres significativos para las variables, puedes hacer que tu código sea más claro y comprensible tanto para ti como para otros programadores que lean o mantengan tu código en el futuro.
6. Facilita el mantenimiento y la depuración: El uso de variables bien definidas y organizadas facilita el mantenimiento y la depuración del código. Si encuentras un error o necesitas realizar cambios en una parte específica del código, es más fácil ubicar y actualizar las variables relacionadas en lugar de rastrear y modificar valores dispersos en todo el programa.

Al igual que en C al declarar las variables reservamos lugar de memoria.

Se declaran de la siguiente manera

`tipoDeDato nombreDeVariable;`

```
// Variables numéricas      int edad;
                             double salario;
                             float peso;

// Variables de texto       String nombre;
                             char inicial;

// Variables booleanas     boolean esMayorDeEdad;

// Variables de referencia a objetos  MiClase objeto;
                                     ArrayList<Integer> listaNumeros;
```

También podemos declarar las variables y asignarles el valor en una sola línea

```
int cantidad = 10;
String mensaje = "Hola, mundo!";
boolean esActivo = true;
```

Operadores

En Java, existen varios operadores que se utilizan para realizar diferentes operaciones en expresiones y manipular valores. A continuación, se presentan los operadores más comunes en Java:

Operadores aritméticos:

+: Suma dos valores.

-: Resta dos valores.

*: Multiplica dos valores.

/: Divide dos valores.

?: Devuelve el resto de la división.

Operadores de asignación:

=: Asigna un valor a una variable.

+=: Asignación con adición.

-=: Asignación con sustracción.

*=: Asignación con multiplicación.

/=: Asignación con división.

%=: Asignación con módulo.

Operadores de incremento/decremento:

++: Incrementa en uno el valor de una variable.

--: Decrementa en uno el valor de una variable.

Operadores de comparación:

==: Comprueba si dos valores son iguales.

!=: Comprueba si dos valores son diferentes.

>: Comprueba si el valor de la izquierda es mayor que el de la derecha.

<: Comprueba si el valor de la izquierda es menor que el de la derecha.

>=: Comprueba si el valor de la izquierda es mayor o igual que el de la derecha.

<=: Comprueba si el valor de la izquierda es menor o igual que el de la derecha.

Operadores lógicos:

&&: Operador AND lógico.

||: Operador OR lógico.

!: Operador NOT lógico.

Operadores de concatenación:

+: Concatena dos cadenas de texto.

Operadores de desplazamiento:

<<: Desplazamiento a la izquierda.

>>: Desplazamiento a la derecha.

Estos son solo algunos ejemplos de operadores en Java. También existen otros operadores, como los operadores bit a bit, los operadores condicionales y los operadores de instancia. Es importante tener en cuenta la precedencia de los operadores al escribir expresiones para asegurar que se evalúen en el orden deseado.

Recuerda que los operadores en Java se utilizan para realizar operaciones en variables y valores, y desempeñan un papel crucial en el desarrollo de programas.