



# Fundamentos de Java



Elementos Básicos

# Contenido

- Variables
- Constantes
- Ámbito de variables
- Operadores
- Casting de variables
- Construcción y manipulación de cadenas
- Leer datos ingresados por Teclado



# Variables

# Variables Primitivas y Referenciales

Tipo de Dato	Representación	Bytes	Rango	Defecto	Clase
byte	Entero	1	-128 a 127	0	Byte
short	Entero	2	-32768 a 32767	0	Short
int	Entero	4	-2147483648 a 2147483647	0	Integer
long	Entero	8	-9223372036854775808 a 9223372036854775807	0	Long
float	Decimal	4	$\pm 3.4 \times 10^{-38}$ a $\pm 3.4 \times 10^{38}$	0.0	Float
double	Decimal	8	$\pm 1.8 \times 10^{-308}$ a $\pm 1.8 \times 10^{308}$	0.0	Double
char	Carácter	2	\u0000 a \uFFFF	\u0000	Character
boolean	Lógico	-	true / false	false	Boolean
void	-	-	-	-	Void

# Constantes

...

```
final double PI = 3.1416;
```

# final / static

## final

- Variable tipo constante.
- No admite cambios después de su declaración y asignación de valor.
- Determina que un atributo no puede ser sobrescrito o redefinido.
- Toda constante declarada con final ha de ser inicializada en el mismo momento de ser declarada.

## static

- Los atributos miembros de una clase pueden ser atributos de clase o atributos de instancia.
- Ocupa un único lugar en memoria.
- Si no se usa static, el sistema crea un lugar nuevo para esa variable con cada instancia (la variable es diferente para cada objeto).
- Cuando usamos “static final” se dice que creamos una constante de clase, un atributo común a todos los objetos de esa clase.

# Ámbito de Variables

## Globales:

- Es un dato accesible en todos los ámbitos de un programa.
- Puede ser modificada en cualquier parte del programa.
- Su uso debe ser analizado con anticipación para el bienestar del programa.

## Locales:

- Es un dato accesible en un ámbito en específico.
- Puede ser modificada en el ámbito en donde fue declarada.
- Su uso depende del objetivo del método.

**Ejemplo**



# Operadores

Nombre	Operadores
Aritméticos	<code>+, -, *, / , %</code>
De relación	<code>&lt; , &gt; , &lt;= , &gt;= , != , ==</code>
Lógicos	<code>&amp;&amp; ó &amp;,    ó  , !</code>
Asignación	<code>++ , -- , = , *= , /= , %= , += , -=</code>
Condicional (Ternario)	<code>() ? :</code>
Prioridad	<code>() , []</code>

# Operadores

```
public class Example {  
  
    public static void main(String[] args) {  
        // Operadores aritméticos  
        System.out.println(20 + 5); // 25  
        System.out.println(10 - 15); // -5  
        System.out.println(20 * 5); // 100  
        System.out.println(60 / 2); // 30  
        System.out.println(60 % 2); // 0  
    }  
}
```

# Operadores

```
public class Example {  
  
    public static void main(String[] args) {  
        // Operadores de relación  
        System.out.println(20 == 5); // false  
        System.out.println(10 > 15); // false  
        System.out.println(20 < 5); // false  
        System.out.println(60 >= 2); // true  
        System.out.println(60 <= 2); // false  
    }  
}
```

# Operadores

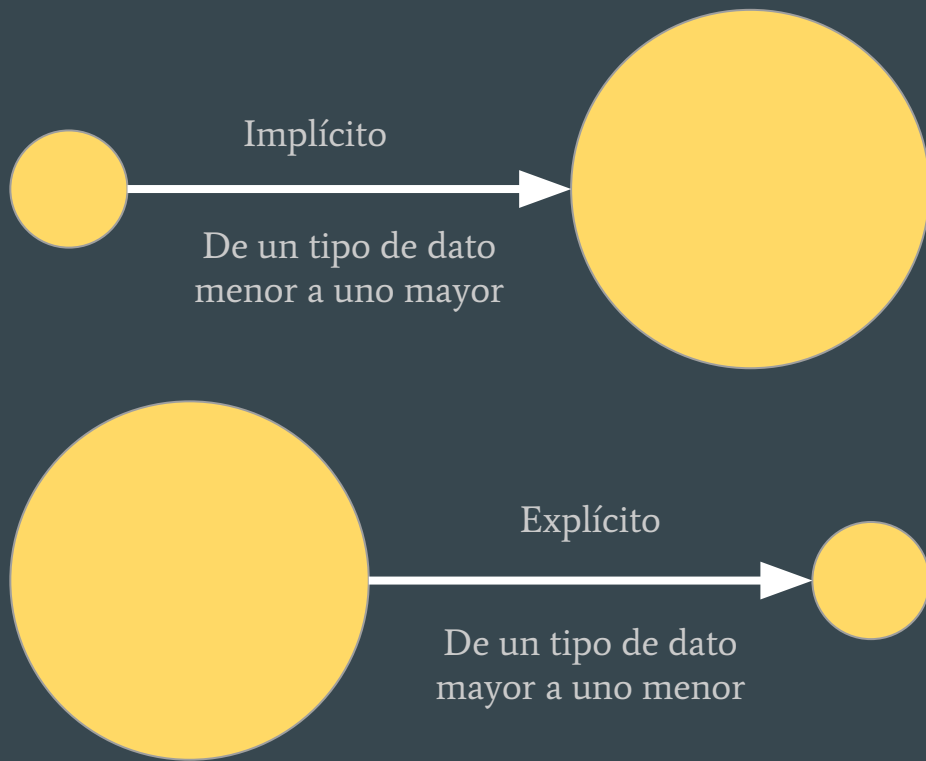
```
public class Example {  
  
    public static void main(String[] args) {  
        // Operadores lógicos  
        System.out.println(true && true); // true  
        System.out.println(true && false); // false  
        System.out.println(true || true); // true  
        System.out.println(true || false); // true  
        System.out.println(!true); // false  
    }  
}
```

**Ejemplo**

# Casting de Variables

- Es una conversión de tipo de dato.
- Sólo aplica a los tipos de dato primitivo.
- Existen dos tipos de casting:
  - Implícito.
  - Explícito.

NOTA: la conversión explícita debe ser usada con cuidado ya que se puede perder información.



# Tabla de Conversión entre Tipos de Datos

Orígen	boolean	byte	short	char	int	long	float	double
boolean	no	no	no	no	no	no	no	no
byte	no	-	si	cast	si	si	si	si
short	no	cast	-	cast	si	si	si	si
char	no	cast	cast	-	si	si	si	si
int	no	cast	cast	cast	-	si	si*	si*
long	no	cast	cast	cast	cast	-	si*	si*
float	no	cast	cast	cast	cast	cast	-	si*
double	no	cast	cast	cast	cast	cast	cast	-

**no:** no conversión. | **sí:** casting es implícito. | **sí\*:** casting es implícito con pérdida de precisión. | **cast:** indica que hay que hacer casting explícito.

**Ejemplo**



# Construcción de Cadenas: String / StringBuffer / StringBuilder

- Se debe entender que:
  - Son inmutables
  - Las cadenas de caracteres son arreglos de caracteres
- Diferencias entre clases:
  - String no permite cambiar el valor de la cadena de caracteres, es síncrona
  - StringBuilder permite cambiar la cadena de caracteres y es síncrona
  - StringBuffer permite cambiar la cadena de caracteres, es asíncrona y además es multihilo

# Manipulación de Cadenas: Clase String

- Se almacena con una instancia
- Son inmutables
- Métodos principales:
  - `length`: devuelve la cantidad de caracteres de la cadena
  - `toUpperCase`: devuelve la cadena convertida a mayúsculas
  - `toLowerCase`: devuelve la cadena convertida a minúsculas
  - `equals`: compara dos cadenas y devuelve `true` si son iguales

# Manipulación de Cadenas

```
public class Example {  
  
    public static void main(String[] args) {  
        // Crear y manipular cadenas de caracteres  
        // Inmutables  
        String cadena1 = "Hola";  
        cadena1 += " Mundo.";  
        // Mutables  
        StringBuilder cadena2 = new StringBuilder();  
        cadena2.append("Otro Hola").append("Mundo");  
  
        System.out.println(cadena1.toUpperCase()); // hola mundo  
        System.out.println(cadena2.equals(cadena1)); // false  
    }  
}
```

**Ejemplo**

# Leer datos por teclado: Scanner

- Aclaración:
  - `System.out`
  - `System.in`
- Clase:
  - `java.util.Scanner`
- Es un tipo de dato object
- Está diseñada para leer los bytes y convertirlo en valores primitivos (int, double, bool, etc) o en valores String
- Entre sus métodos principales:
  - `nextByte()`
  - `nextDouble()`
  - `nextFloat()`
  - `nextInt()`
  - `next()`
  - `nextLine()`
  - `nextLong()`

**Ejemplo**