

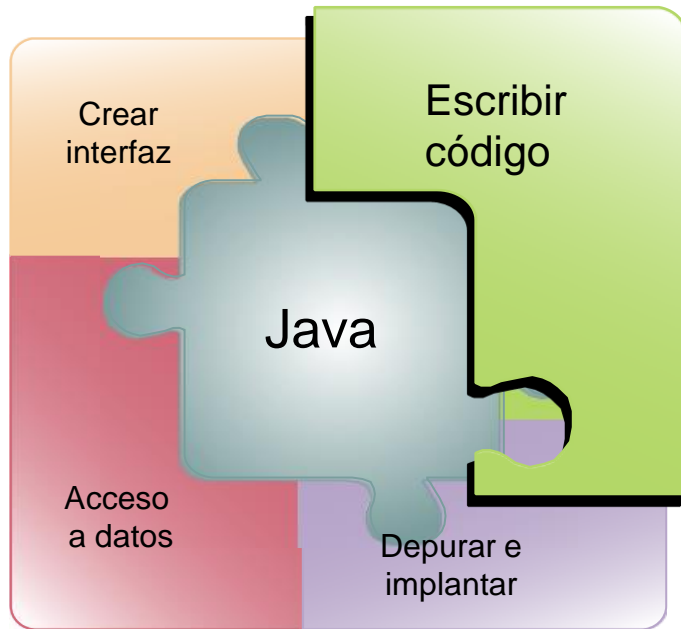
---

# Elementos del lenguaje

## ■ Unidad 2

# Descripción

---



1. Estructura básica de un programa
2. Variables y tipos de datos
3. Literales
4. Convertir tipos de datos
5. Operadores
6. Entrada /Salida
7. Tipos enumerados
8. Ejercicios
9. Para ampliar...



**estructura básica**

**comentarios y separadores**

# **ESTRUCTURA BÁSICA DE UN PROGRAMA**

# Estructura básica de un programa

---

```
/*
 * Estructura de una clase de Java
 * Si no es una clase principal el método main no aparece
 */

public class NombreDeLaClase {
    //Declaración de los atributos de la clase

    //Declaración de los métodos de la clase

    //El método main que indica dónde empieza la ejecución
    public static void main (String[] args) {
        //Declaración de las variables del método

        //Sentencias de ejecución del método
    }
}

//Si no es una clase principal el método main no aparece
```

# Comentarios y separadores

---

## ■ Comentarios:

- `/* ... diversas líneas ... */`
- `// ....` Una sola línea

## ■ Separadores:

- `(...)` Listas de parámetros en la definición y llamada a un método
- `{...}` Engloba bloques de código y en valores iniciales de arrays
- `[...]` En la declaración de arrays y en referencias a elementos de éstos
- `;` Separador de instrucciones
- `,` Separador de identificadores y argumentos
- `.` Separador de elementos de un objeto



**variables y Tipos**

**tipos de datos**

**tipos de datos primitivos**

**identificadores de variables**

**declaración de variables**

**inicialización de variables**

**asignación**

**ámbito**

**constantes**

# **VARIABLES Y TIPOS DE DATOS**

# Variables y Tipos de datos

---

- Una variable permite almacenar los datos, resultados y resultados intermedios de un problema en un programa
- Tiene asociado un tipo de datos
- El tipo de datos al que pertenece una variable:
  - define el conjunto de valores que son susceptibles de ser almacenados en dicha variable
  - y las operaciones que se pueden realizar con ella
- Declaración de variables en Java:  
*tipo identificador ;*

# Tipos de datos

---

- **En Java, los tipos de datos pueden clasificarse en dos grupos:**

- Tipo de datos primitivos
- Tipo de datos referencia:  
Strings, Arrays, Clases e Interfaces  
Los estudiaremos más adelante...

- **Más información sobre los tipos de datos básicos de Java :**

- [www.aprenderaprogramar.com](http://www.aprenderaprogramar.com)
- [puntocomnoesunlenguaje.blogspot.com.es](http://puntocomnoesunlenguaje.blogspot.com.es)



# Tipos de datos primitivos

---

## ■ Tipos Numéricos

NOMBRE	TAMAÑO EN BITS	VALOR MÁXIMO
byte	8	127
short	16	32767
int	32	2147483647
long	64	9223372036854775807
float	32	3.4E38
double	64	1.7E+308

# Tipos de datos primitivos. Tipos Numéricos

---

- **byte:** El tipo de dato byte es un entero de 8 bits. Su valor mínimo es -128 y el máximo 127 (inclusive)
- **short:** El tipo de dato short es un entero de 16 bits. Su valor mínimo es -32,768 y el máximo 32,767 (inclusive)
- **int:** El tipo de dato int es un entero de 32 bits. Su valor mínimo es -2,147,483,648 y el máximo 2,147,483,647 (inclusive). Generalmente este tipo es la elección predeterminada para valores enteros
- **long:** El tipo de dato long es un entero de 64 bits complemento a dos. Su valor mínimo es -9,223,372,036,854,775,808 y el máximo 9,223,372,036,854,775,807 (inclusive)
- **float:** El tipo de dato float es un dato en coma flotante IEEE 754 de 32 bits y precisión simple
- **double:** El tipo de dato double es un dato en coma flotante IEEE 754 de 64 bits y precisión doble. Normalmente este tipo de dato es la elección predeterminada para valores decimales

# Tipos de datos primitivos

---

## ■ Tipo carácter

Nombre	Tamaño	Representa
char	2 Bytes	Representa caracteres como letras, números y caracteres especiales Java utiliza la codificación Unicode de 2 bytes

## ■ Tipo booleano o lógico

Nombre	Tamaño	Representa
boolean	1 Byte	Puede tomar los valores true/false

# Identificadores de variables

---

- Los identificadores son los nombres que se les da a las variables, clases, interfaces, atributos y métodos de un programa
- Reglas para la creación de identificadores:
  - Java hace distinción entre mayúsculas y minúsculas:
    - var1, Var1 y VAR1 son distintos
  - La longitud máxima de los identificadores es prácticamente ilimitada
  - No puede ser una palabra reservada del lenguaje ni los valores lógicos true o false
  - No pueden ser iguales a otro identificador declarado en el mismo ámbito
  - Debe empezar por una letra y seguir con una sucesión de letras y dígitos. Se considera letra caracteres latinos, hebreos, cirílicos, etc... También '\$' (no utilizar) y '\_' (subrayado)
  - No pueden utilizarse espacios en blanco ni símbolos coincidentes con operadores

# Identificadores de variables II

---

- Convención:

- los nombres de las variables y los métodos deberían empezar por una letra minúscula y los de las clases por mayúscula

- Si el identificador está formado por varias palabras, la primera se escribe en minúsculas (excepto para las clases) y el resto de palabras se empiezan por mayúscula

variable: añoDeCreación

clase: HolaMundo

- NOTA: Tenéis en el material complementario un documento oficial de Sun traducido al castellano sobre Convenciones de codificación

# Declaración de variables

---

- **Para declarar una variable:**

```
tipo identificador;
```

```
tipo identificador = valor;
```

```
tipo ident1, ident2, ident3, etc...;
```

```
tipo ident1=valor1, ident2=valor2, etc...;
```

- **Por ejemplo:**

```
int edadPedro=34;
```

```
float precioPatata=1.2F, precioChoco=2.3F;
```

```
char car='A', car2='\u0041';
```

```
// 0041 es el código Unicode en hexadecimal de  
la A mayúscula
```

# Inicialización de variables

---

- **Si una variable no ha sido inicializada tiene un valor asignado por defecto:**
  - Para las variables de tipo numérico, el valor por defecto es cero ( 0 )
  - Las variables de tipo char, el valor '\u0000'
  - Las variables de tipo boolean, el valor false
  - Para las variables de tipo referencial (objetos), el valor null

# Asignación

---

- La instrucción de asignación permite asignar valores a las variables o modificar los que ya tienen
- Su sintaxis es:  
identificador = expresión;
- Se puede utilizar en el bloque de declaración de variables para definir valores iniciales:

```
int suma=64;  
char ch1, ch2='u';  
float f1=2.0F, f2;  
.....  
f2=34.67F;  
suma=suma+2;
```



# Ámbito de una variable

---

- Se llama **ámbito de una variable** a la parte del programa en la que es conocida y se puede utilizar
- Una variable local se declara dentro del cuerpo de un método de una clase y es visible únicamente dentro de dicho método
- Se puede declarar en cualquier lugar del cuerpo, incluso después de instrucciones ejecutables, aunque es una buena costumbre declararlas justo al principio
- También pueden declararse variables dentro de un bloque parentizado por llaves { ... }
  - Sólo serán “visibles” dentro de dicho bloque
  - Las variables definidas en un bloque deben tener nombres diferentes

# Constantes

---

- Para declarar una constante usamos el modificador *final*

final double PI = 3.1415926536;

- El valor de una constante no se puede modificar durante el programa
- Debemos darle un valor a la vez que se declara



**LITERALES**

# Literales I

---

- **Un literal es un valor que se expresa a sí mismo**
- **Literal entero puede expresarse:**
  - en decimal (base 10)  
Ejemplo: 21
  - octal (base 8)  
Ejemplo: 025
  - hexadecimal (base 16)  
Ejemplo: x03A
  - Por defecto el literal es int. Puede añadirse al final del mismo la letra L ó l para indicar que el entero es considerado como long
- **Literal real puede expresarse:**
  - parte entera seguida del punto decimal ( . ) y la parte fraccionaria ( Ej: 345.678 - 0.00056 )
  - notación exponencial o científica ( Ej: 3.45678e2 - 5.6e-4 )
  - Se puede poner una letra como sufijo:
    - F ó f      Trata el literal como de tipo float
    - D ó d      Trata el literal como de tipo double
  - Por defecto el literal es double. Si deseamos que se interprete como float debemos añadir el sufijo F

# Literales II

## ■ Literal carácter

Puede escribirse como:

- **Un carácter entre comillas simples como 'a', 'ñ', 'Z', 'p', etc.**
- **El código Unicode del carácter:**  
97 es el código Unicode del carácter a  
65 es el código Unicode del carácter A
- **Entre comillas, anteponiendo la secuencia de escape '\'** si el código Unicode lo expresamos en octal  
'\141'                                      código Unicode en **octal** equivalente a 'a'  
'\101'                                      código Unicode en **octal** equivalente a 'A'
- **Entre comillas, anteponiendo la secuencia de escape '\u'** si código Unicode lo expresamos en hexadecimal  
'\u0061'                                      código Unicode en **hexadecimal** equivalente a 'a'  
'\u0041'                                      código Unicode en **hexadecimal** equivalente a 'A'

Existen unos caracteres especiales que se representan utilizando secuencias de escape:

<u>Secuencia</u>	<u>Significado</u>
\'	Comilla simple
\"	Comillas dobles
\\	Contrabarra
\b	Backspace
\n	Cambio de línea
\r	Retorno de carro
\t	Tabulador
\f	Salto de página

# Literales III

---

## ■ Literal booleano:

- palabras reservadas true y false

Ejemplo: boolean activado = false;

## ■ Literal Strings o cadena de caracteres

- No forman parte de los tipos de datos elementales en Java

- Encerrado entre comillas dobles ( " )

Ejemplo:

```
System.out.println("Primera línea\nSegunda línea del string");
```

```
System.out.println("Ho\u0061");
```

```
System.out.println("Escribe \n y no saltes de línea");
```

---

## **4.- CONVERTIR TIPOS DE DATOS**

# Conversión de tipos

---

- Cuando se realiza una instrucción de asignación:

Identificador = expresión;

**tanto la variable como la expresión deben de ser del mismo tipo o de tipos compatibles**

- Una expresión puede asignarse a una variable siempre que sea de un tipo de tamaño menor que el tipo de la variable. Por lo tanto podemos asignar en este orden:

short ➡ int ➡ long ➡ float ➡ double

- Otras formas de conversión de tipos se pueden realizar explícitamente a través de lo que se llama casting


(tipo) expresión

Ejemplo:

num= (int) 34.56

- También utilizando funciones adecuadas de ciertos paquetes





**operadores unarios**  
**operadores aritméticos**  
**operadores de comparación**  
**operadores lógicos**  
**combinar operadores lógicos y de comparación**  
**operadores de asignación**  
**operadores de concatenación**  
**operadores de bit ( ...ya lo veremos)**  
**prioridad de operadores**

## **5.- OPERADORES**

# Operadores aritméticos I

---

- Pueden realizar operaciones aritméticas que implican el cálculo de valores numéricos representados por literales, variables, otras expresiones, llamadas de funciones y propiedades, y constantes

- **Sintaxis:**

```
expresion1 operador_aritmético expresion2
```

- **Ejemplo:**

```
int x;  
x = 52 * 17;  
x = 120 / 4;  
x = 67 + 34;  
x = 32 - 12;  
X = 7 % 2;
```

# Operadores aritméticos II

---

**+**      **Suma**

**-**      **Resta**

**\***      **Multiplicación**

**/**      **División**

**%**      **Resto de la división entera**

# Operadores Unarios

---

## ■ Signo

- Poner un signo + o un signo - delante de una expresión
- Ejemplo:  
+45  
-32

## ■ Incremento (++) y Decremento (--)

- Aumentar y disminuir en 1 el valor de la variable
- Pueden ir delante(pre) o detrás(post) de la variable
- Ejemplo:  
int valor, i=5;  
i++; // ahora i vale 6. Es equivalente a i=i+1;  
--i; // ahora i vale 5. Es equivalente a i=i-1;
- La diferencia entre pre y post aparece en una instrucción compuesta:  
valor=i++; //ahora valor vale 5 y i vale 6  
// Es equivalente a { valor=i; i=i+1; }  
valor=++i; // ahora valor vale 7 y i vale 7  
// Es equivalente a { i=i+1; valor=i; }

# Operadores de comparación I

---

- Símbolos que evalúan expresiones condicionales y devuelven un valor boolean

- **Sintaxis:**

```
expresion1 operador_de_comparación expresion2
```

# Operadores de comparación II

---

Operador	True si	False si
< (Menor que)	<code>expresion1 &lt; expresion2</code>	<code>expresion1 &gt;= expresion2</code>
<= (Menor o igual que)	<code>expresion1 &lt;= expresion2</code>	<code>expresion1 &gt; expresion2</code>
> (Mayor que)	<code>expresion1 &gt; expresion2</code>	<code>expresion1 &lt;= expresion2</code>
>= (Mayor o igual que)	<code>expresion1 &gt;= expresion2</code>	<code>expresion1 &lt; expresion2</code>
= (Igual a)	<code>expresion1 == expresion2</code>	<code>expresion1 != expresion2</code>
!= (Distinto de)	<code>expresion1 != expresion2</code>	<code>expresion1 == expresion2</code>

# Operadores de comparación III

---

## ■ Ejemplo:

```
int  cantidad=3000;  
boolean pedidoGrande;  
pedidoGrande = cantidad > 1000
```

```
boolean testResult ;  
testResult = ( 45 < 35 );  
testResult = ( 45 == 45 );  
testResult = ( 4 != 3 );  
testResult = ( 'a' > 'b' );
```

# Operadores lógicos I

---

- Los operadores lógicos realizan una evaluación lógica de expresiones y devuelven un valor boolean

- **Sintaxis:**

```
expresion1 operador_lógico expresion2
```

- **Ejemplo:**

```
edad>18 && sexo=='H'
```

```
edad>18 || sexo=='H'
```



# Operadores lógicos II

---

Operador	Función
&&	Combina dos expresiones. Cada expresión debe ser <b>True</b> para que toda la expresión sea <b>True</b>
	Combina dos expresiones. Si una expresión es <b>True</b> , toda la expresión es <b>True</b> .
!	Proporciona el negativo lógico de la entrada

# Operadores lógicos III . Tablas de verdad

Operador lógico **And**

<u>Expres1</u>	<u>Expres2</u>	<u>Resultado</u>
True	True	True
True	False	False
False	True	False
False	False	False

Operador lógico **Not**

<u>Expresión1</u>	<u>Resultado</u>
True	False
False	True

Operador lógico **Or**

<u>Expres1</u>	<u>Expres2</u>	<u>Resultado</u>
True	True	True
True	False	True
False	True	True
False	False	False

# Combinar operadores lógicos y de comparación

---

- Podemos combinar operadores de comparación y operadores lógicos con instrucciones condicionales
- Ejemplo:

Operador de comparación

Operador lógico

```
edad <= 25 && edad >=14
```

# Operadores de asignación

---

Operador	ejemplo
<b>=</b>	<b>edad = 34</b>
<b>+=</b>	<b>edad += 1</b> <b>edad = edad +1</b>
<b>- =</b>	<b>edad - = 3</b> <b>edad = edad – 3</b>
<b>* =</b>	<b>edad *= 2</b> <b>edad = edad * 2</b>
<b>/=</b>	<b>edad /= 2</b> <b>edad = edad / 2</b>
<b>% =</b>	<b>edad %= 2</b> <b>edad = edad % 2</b>

# Operador de concatenación

---

- **Permite generar una cadena de caracteres a partir de otras dos**

**expresion1 + expresión2**

Ejemplo: **“Hola” + “, “ + ”buenos días”**

Ejemplo: **“Debe pagar :” + total + ” euros”**

# Prioridad de operadores I

---

- Cuando aparecen varias operaciones en una expresión se evalúa y se resuelve en un orden predeterminado
- Orden por niveles:
  1. **Paréntesis, de dentro a fuera**
  2. **Unarios**
  3. **Aritméticos**
    - A. Multiplicativos:     \*             /             %
    - B. Sumativos:           +             -
  4. **Comparativos o Relacionales**
  5. **Lógicos o booleanos**
    - A. Not     !
    - B. And     &&
    - C. Or     ||
  6. **Asignación**
- Cuando aparecen operadores de la misma prioridad juntos en una expresión el compilador evalúa cada operación de izquierda a derecha

# Prioridad de operadores II

---

## ■ Ejemplo

$a = -3 + 5 + 2 * 4 - 6 / 4 * 3 - 5 \% 2;$

ORDEN: 1 6 7 2 8 3 4 9 5

**Valor final a igual a 6**

## ■ Ejemplo

`System.out.println ("Cuidado!!!: " + 4*5 + 6 );`



**Salida de datos por pantalla**

**Entrada de datos del teclado**

## **6.- ENTRADA / SALIDA**



# Salida de datos por pantalla

---

- Utilizaremos los métodos `print( )` o `println( )`
  - `println( )` incluye el retorno de carro al final de la salida

```
System.out.print("Se imprime este mensaje sin el  
retorno de carro");
```

```
System.out.println("Se imprime este mensaje con un  
retorno de carro");
```

# Entrada de datos del teclado I

---

- **El método `read()` lee un solo carácter**

```
char c = (char) System.in.read();
```

- Esta manera de leer del teclado es muy poco práctica y sería tedioso programar la lectura de un número, de una cadena de caracteres... (no la usaremos)

- **Declarar un objeto de la clase `Scanner` y usar sus métodos**

- Entenderemos los detalles en próximos temas
- En el ejemplo siguiente vemos como hacerlo

# Entrada de datos del teclado II

---

**//1.-Importamos el fichero donde están las clases**

**import java.util.Scanner;**

public class Exemple {

public static void main (String[] args) {

int primerNum;

**//2.-Se declara el objeto lector de la clase Scanner**

**Scanner lector = new Scanner(System.in);**

...

System.out.print("Escribe un número y pulsa retorn: ");

//se lee un valor entero

**primerNum = lector.nextInt();**

System.out.print("El numero introducido es: "+primerNum);

...

}

}

# Entrada de datos del teclado III

---

<u>Método</u>	<u>Tipo de dato leído</u>
<code>lector.nextByte()</code>	<code>byte</code>
<code>lector.nextShort()</code>	<code>short</code>
<code>lector.nextInt()</code>	<code>int</code>
<code>lector.nextLong()</code>	<code>long</code>
<code>lector.nextFloat()</code>	<code>float</code>
<code>lector.nextDouble()</code>	<code>double</code>
<code>lector.nextBoolean()</code>	<code>boolean</code>
<code>lector.next()</code>	<code>String</code>
<code>Lector.nextLine()</code>	<code>String</code>

# Ejemplo I

---

```
import java.util.Scanner;

/*
 *ejemplo Scanner
 */
public class EjemploScanner {

    public static void main(String[] args) {
        int entero;
        double real;
        boolean siOno;
        char caracter;
        String texto; //no es un tipo primitivo

        Scanner lector = new Scanner(System.in);

        System.out.print("Introduce un entero: ");
        entero = lector.nextInt();
        System.out.println("El entero introducido es " + entero);

        System.out.print("Introduce un real: ");
        real = lector.nextDouble();
        //ATENCIÓN! el programa fallara si el usuario entra el numero con punto decimal
        System.out.println("El real introducido es " + real);

        System.out.print("Introduce si o no: ");
        siOno = lector.nextBoolean();
        //ATENCIÓN! el programa fallara si el usuario no entra true o false
        System.out.println("El boolean introducido es " + siOno);
    }
}
```

# Ejemplo I

---

```
System.out.print("Introduce si o no: ");
siOno = lector.nextBoolean();
//ATENCIÓN! el programa fallara si el usuario no entra true o false
System.out.println("El boolean introducido es " + siOno);

lector.nextLine();
//ATENCIÓN! con esta instrucción limpio el buffer de entrada
//si no la pongo, no me deja entrar el carácter posterior
//encuentra el salto de linea y considera que esa es la tira introducida
//pruébalo!

System.out.print("Introduce un caracter: ");
caracter = lector.nextLine().charAt(0);
//ATENCIÓN! leo un String y me quedo con el primer carácter
System.out.println("El caracter introducido es " + caracter);

//lector.nextLine();
//Hace falta aquí limpiar el buffer de entrada?

System.out.print("Introduce tu nombre: ");
texto = lector.nextLine();
System.out.println("Tu nombre es " + texto);

}
```

# Ejemplo II

---

```
import java.util.*;
Public class Adivinanza {
    public static void main (String[] args) {
        Scanner tcl = new Scanner(System.in);
        int valor;
        System.out.println("Piensa un numero"); tcl.nextLine();
        System.out.println("Multiplicalo por 5");tcl.nextLine();
        System.out.println("Sumale 6"); tcl.nextLine();
        System.out.println("Multiplicalo por 4");tcl.nextLine();
        System.out.println("Sumale 9"); tcl.nextLine();
        System.out.println("Multiplicalo por 5");tcl.nextLine();
        System.out.println("Escribe el resultado");
        valor=tcl.nextInt();
        System.out.println("El numero que has pensado es: ");
        System.out.println((valor-165)/100);
    }
}
```

---

## **7.-TIPOS ENUMERADOS**



# Tipos Enumerados

- **Son conjuntos de valores constantes para los que no existe un tipo predefinido**
  - Por ejemplo: para representar los días de la semana, estaciones del año, meses del año, etc...
- **Se implementa de la siguiente manera:**

```
public class Semana {
    public enum DiaSemana{LUNES, MARTES, MIERCOLES, JUEVES,
                          VIERNES, SABADO, DOMINGO}

    public static void main (String[] args){

        DiaSemana hoy = DiaSemana.JUEVES;
        DiaSemana ultimo=DiaSemana.DOMINGO;

        System.out.println( "Hoy es " +hoy +"\n Y el ultimo dia es
                               "+ultimo );
    }
}
```


---

## **8.- EJERCICIOS**

# Ejercicio:

---

- **Escribe un programa que permita introducir dos números y calcular la suma de los mismos**
- **Provoca errores en el programa:**
  - Errores de compilación:
    - No declarar variables
    - Asignar tipos
    - ....
  - Errores de ejecución:
    - Introducir un string
    - Introducir un entero demasiado grande
    - ...



la clase Math  
el tipo String  
sistema de tipos

## **9.- PARA AMPLIAR...**

# Algunas funciones predefinidas. La clase Math

---

## ■ Las constantes E y PI

`Math.E=2.7182818284590452354`

`Math.PI=3.14.159265358979323846`

## ■ Las funciones de redondeo, con x de tipo double:

- `ceil(x)` : devuelve el número entero más pequeño que es mayor o igual a x
- `floor(x)` : devuelve el número entero más grande que es menor o igual a x
- `round(x)` : convierte el real x al entero más próximo

## ■ Funciones trigonométricas

- `sin(x)` : calcula el seno del ángulo (en radianes) x
- `cos(x)` : calcula el coseno del ángulo (en radianes) x
- `asin(x)` : calcula el arco seno de x (x entre -1 y 1)
- `acos(x)` : calcula el arco coseno de x (x entre -1 y 1)
- `atan(x)` : calcula el arco tangente de x

# Algunas funciones predefinidas. La clase Math

---

## ■ Otras funciones

- $\text{abs}(x)$  : calcula el valor absoluto de  $x$  (entero o real)
- $\text{exp}(x)$  : calcula  $e$  elevado a  $x$  ( $x$  es real)
- $\text{log}(x)$  : calcula el logaritmo natural de  $x$  ( $x$  real y no negativo)
- $\text{max}(x,y)$  : compara los números  $x$  e  $y$  (enteros o reales) y devuelve el mayor
- $\text{min}(x,y)$  : compara los números  $x$  e  $y$  (enteros o reales) y devuelve el menor
- $\text{pow}(x,y)$  : calcula  $x$  elevado a  $y$ . No está definida si  $x$  es negativo o 0 e  $y$  no es entero, ni tampoco si  $x=0$  e  $y$  es negativo o 0
- $\text{random}()$  : genera un número pseudo-aleatorio entre 0.0 y 1.0
- $\text{sqrt}(x)$  : calcula la raíz cuadrada de  $x$  ( $x$  no negativo)

# Tipo String. Uso sencillo

---

- El tipo String permite representar secuencias de caracteres
- Ejemplo:

```
String frase, palabra, linea;
```

```
frase="En un lugar de la Mancha de cuyo nom...";
```

```
Scanner lector = new Scanner(System.in);
```

```
//.....solicitamos al usuario un texto
```

```
//..... y el texto de entrada es: Oh! es terrible
```

```
palabra=lector.next( ); //palabra = Oh!
```

```
linea=lector.nextLine( ); // linea = Oh! es terrible
```

# Sistema de tipos

---

- **Dos categorías generales de tipos:**

- ***tipo simple o valor***

- Una variable de tipo valor contiene directamente sus datos
- Cada variable de tipo valor tiene su propia copia de datos, de modo que las operaciones en una variable de tipo valor no pueden afectar a otra variable

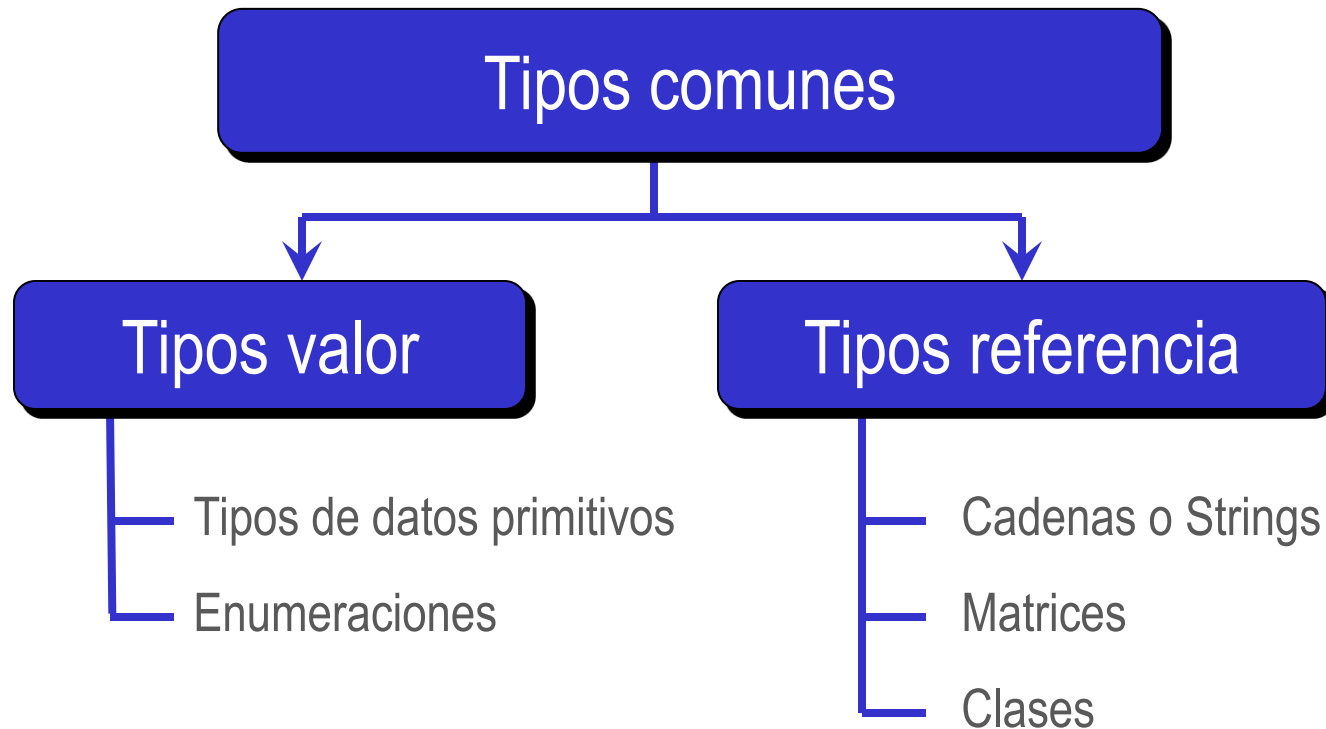
- ***tipo referencia***

- Una variable de tipo referencia contiene una referencia o puntero al valor de un objeto
- Dos variables de tipo referencia pueden referirse al mismo objeto, de modo que las operaciones en una variable de tipo referencia pueden afectar al objeto referenciado por otra variable de tipo referencia



# Sistema de tipos

---



# Sistema de tipos

---

Los tipos de datos simples pueden ser declarados como referenciales (objetos) ya que existen clases que los engloban

## Tipos de datos simples

byte

short

int

long

float

double

char

boolean

## Clase equivalente

java.lang.Byte

java.lang.Short

java.lang.Integer

java.lang.Long

java.lang.Float

java.lang.Double

java.lang.Character

java.lang.Boolean