

Revisión de conceptos básicos de programación

Contenido

- Tipos simples, expresiones y variables
- Estructuras de control de flujo. Selección e iteración
- Tipos estructurados. Vectores y cadenas
- Uso de clases y objetos
- Métodos y parámetros
- Entrada/salida. Archivos y flujos de datos

El «Hola mundo» en Python

hola_mundo.py

```
print('¡Hola, mundo!')
```

El «Hola mundo» en Java

HolaMundo.java

El nombre del archivo debe coincidir con el nombre de la clase

```
public class HolaMundo {  
    public static void main(String[] args) {  
        System.out.println("¡Hola, mundo!");  
    }  
}
```

Un programa Java está formado por una o más clases relacionadas

El «Hola mundo» en Java

HolaMundo.java

```
public class HolaMundo {  
    public static void main(String[] args) {  
        System.out.println("¡Hola, mundo!");  
    }  
}
```

Programa principal

Lenguaje de «formato libre»

TablasMultiplicar.java

```
public class TablasMultiplicar {  
    public static void main(String[] args) {  
        for (int i = 1; i <= 10; i++) {  
            System.out.println("\nTabla del " + i);  
            for (int j = 1; j <= 10; j++)  
                System.out.println(i + " x " + j + " = " + i * j);  
        }  
    }  
}
```

Este programa muestra las diez primeras tablas de multiplicar

Lenguaje de «formato libre»

TablasMultiplicarIlegible.java

```
public class TablasMultiplicarIlegible {  
    public static void main(String[] args) {  
        for (int i = 1; i <= 10; i++) {System.  
            out.  
            println("\nTabla del "  
                + i); for (int j =  
                    1; j <= 10; j++)  
                    System.out.println(i + " x " + j + " = " + i * j);  
    }}}
```

Final de línea y sangrado irrelevantes
(el código se comporta igual)

Aunque afectan a la
claridad del código

Lenguaje de «formato libre»

TablasMultiplicar.java

```
public class TablasMultiplicar {  
    public static void main(String[] args) {  
        for (int i = 1; i <= 10; i++) {  
            System.out.println("\nTabla del " + i);  
            for (int j = 1; j <= 10; j++)  
                System.out.println(i + " x " + j + " = " + i * j);  
        }  
    }  
}
```

Instrucciones simples
acabadas en punto y coma

Lenguaje de «formato libre»

TablasMultiplicar.java

```
public class TablasMultiplicar {  
    public static void main(String[] args) {  
        for (int i = 1; i <= 10; i++) {  
            System.out.println("\nTabla del " + i);  
            for (int j = 1; j <= 10; j++)  
                System.out.println(i + " x " + j + " = " + i * j);  
        }  
    }  
}
```

Bloques delimitados por llaves

Lenguaje de «formato libre»

TablasMultiplicarErroneo.java

```
public class TablasMultiplicarErroneo {  
    public static void main(String[] args) {  
        for (int i = 1; i <= 10; i++)  
            System.out.println("\nTabla del " + i);  
        for (int j = 1; j <= 10; j++)  
            System.out.println(i + " x " + j + " = " + i * j);  
    }  
}
```

Versión errónea

Comentarios

IndiceMasaCorporal.java

Delimitado por /* y */

```
public class IndiceMasaCorporal {  
    public static void main(String[] args) {  
        /* Aún no sabemos leer de la entrada estándar, por lo que  
           usamos unos valores concretos para el peso y la altura. */  
        double altura = 1.85;  
        double peso = 72;  
        double imc = peso / (altura * altura); // Cálculo del IMC  
        System.out.println("El IMC es " + imc);  
    }  
}
```

Desde // hasta
final de línea

Ejemplo: sumar divisores propios

SumarDivisoresPropios.java

```
public class SumarDivisoresPropios {  
    public static void main(String[] args) {  
        // Aquí escribiríamos las instrucciones necesarias  
    }  
}
```

Nos centraremos
en este código

Ejemplo: sumar divisores propios

SumarDivisoresPropios.java

```
int número = 24; // De momento usamos un valor concreto
// Sumar los divisores propios del número
int sumaDivisores = 0;
int candidatoDivisor = 1;
while (candidatoDivisor <= número / 2) {
    if (número % candidatoDivisor == 0) // Divisor
        sumaDivisores += candidatoDivisor;
    candidatoDivisor += 1;
}
// Falta mostrar el resultado
```

Hay que «declarar» las variables

Algunos tipos de datos «primitivos»

ejemplo

```
int comensales = 4;
```

```
double precioMenú = 5.7;
```

```
boolean conBebida = false;
```

```
char categoría = 'A';
```

Entero

-2^{31} a $2^{31}-1$

Flotante

15 dígitos significativos

Lógico

false y true

Carácter

Algunos tipos de datos «referencia»

ejemplo

```
String reservadoPor = "Pepe";
```

Cadena

```
boolean[] ocupado = new boolean[7];
```

Vector

```
double[][] precios = new double[E][D];
```

Matriz

```
Scanner entrada;
```

Otra clase

Lectura de datos: clase Scanner

TablaMultiplicar.java

```
// Solo se muestra el programa principal
public static void main(String[] args) {
    Scanner entrada = new Scanner(System.in);
    System.out.print("Introduce un número entero: ");
    int número = entrada.nextInt();
    System.out.println("Tabla del " + número);
    for (int i = 1; i <= 10; i++)
        System.out.println(número + " x " + i + " = " + número * i);
}
```

System.in es la
entrada estándar

Lectura de números enteros

TablaMultiplicar.java

```
// Solo se muestra el programa principal
public static void main(String[] args) {
    Scanner entrada = new Scanner(System.in);
    System.out.print("Introduce un número entero: ");
    int número = entrada.nextInt();
    System.out.println("Tabla del " + número);
    for (int i = 1; i <= 10; i++)
        System.out.println(número + " x " + i + " = " + número * i);
}
```

Para leer enteros usamos `nextInt`

Lectura de números con parte decimal

IndiceMasaCorporal.java

```
// Solo se muestra el programa principal
public static void main(String[] args) {
    Scanner entrada = new Scanner(System.in);
    System.out.print("Introduce la altura (en metros): ");
    double altura = entrada.nextDouble();
    System.out.print("Introduce el peso (en kilos): ");
    double peso = entrada.nextDouble();
    double imc = peso / (altura * altura);
    System.out.println("El IMC es " + imc);
}
```

Para leer flotantes usamos **nextDouble**

Lectura de cadenas

LongitudCadena.java

```
// Solo se muestra el programa principal
public static void main(String[] args) {
    Scanner entrada = new Scanner(System.in);
    System.out.print("Introduce una palabra: ");
    String palabra = entrada.next();
    System.out.println("Número de caracteres: " + palabra.length());
}
```

Para leer una «palabra» usamos **next**

Lectura de cadenas

Saludo.java

```
// Solo se muestra el programa principal
public static void main(String[] args) {
    Scanner entrada = new Scanner(System.in);
    System.out.print("Introduce tu nombre: ");
    String nombre = entrada.nextLine();
    System.out.println("¡Hola, " + nombre + "!");
}
```

Para leer una cadena
(hasta final de línea)
usamos **nextLine**

Palabras reservadas

Palabras reservadas en Java				
abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

Palabras reservadas

Las que iremos
estudiando

Palabras reservadas en Java				
abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

Expresiones: operadores

Categoría	Operadores
Aritméticos	+ - * / %
De igualdad	== !=
Relacionales	< <= > >=
Lógicos	&& !
Asignación	= += -= *= /= %=
Incremento	++ --
Condicional	? :
Conversión	(<i>tipo</i>)

Operadores aritméticos

Operador	Significado	Ejemplo	Resultado
+	Suma	4 + 6	10
		5.0 + 3	8.0
-	Resta	20 - 30	-10
-	Cambio de signo	-(20 - 30)	10
*	Multiplicación	5 * 3.5	17.5
/	División	31 / 4	7
		31.0 / 4	7.5
%	Resto división	31 % 4	3

Operadores de igualdad

Operador	Significado	Ejemplo	Resultado
<code>==</code>	Igual que	<code>100 == 2 * 50</code> <code>'A' == 'a'</code> <code>(1 + 1 == 2) == true</code>	<code>true</code> <code>false</code> <code>true</code>
<code>!=</code>	Distinto de	<code>18 % 2 != 0</code> <code>'A' != 'a'</code> <code>(2 != 2) != true</code>	<code>false</code> <code>true</code> <code>true</code>

Más adelante veremos cómo se comportan con tipos referencia

Operadores relacionales

Operador	Significado	Ejemplo	Resultado
<	Menor que	'A' < 'Z'	true
<=	Menor o igual que	10 <= 2 * 5	true
>	Mayor que	-1 > 0	false
>=	Mayor o igual que	'A' >= 'B'	false

No son válidas expresiones
Python como, por ejemplo:

10 < 20 < 30

Operadores lógicos

Operador	Significado
&&	«and»
	«or»
!	«not»



Se evalúan por
«circuito corto»

```
// No triángulo
!(a + b > c && b + c > a && a + c > b)

// No triángulo (otra versión)
a + b <= c || b + c <= a || a + c <= b
```

Operadores de asignación

Operador	Ejemplo
=	área = base * altura
+=	suma += nota
-=	índice -= 1
*=	factorial *= número
/=	número /= 10
%=	minutos %= 60

Operadores de asignación

ejemplo

```
int a = 10, b, c = 20;  
  
System.out.println( a = 15 );  
  
System.out.println(a);  
  
System.out.println( a = b = c );
```

Asigna 15 a la variable a
y se evalúa a 15

Asociativos «a la derecha»

Operadores de conversión

ejemplo

```
int suma = 0, cantidad = 0;  
// Instrucciones para ir leyendo valores, así como  
// ir actualizando suma y cantidad  
double media = (double) suma / cantidad;
```

El operando izquierdo de la
división es un valor flotante

Operador condicional

ParImpar.java

```
import java.util.Scanner;  
public class ParImpar {  
    public static void main(String[] args) {  
        Scanner entrada = new Scanner(System.in);  
        System.out.print("Dime un número entero: ");  
        int n = entrada.nextInt();  
        System.out.println("Es " + (n % 2 == 0 ? "par" : "impar"));  
    }  
}
```

Tiene este aspecto:
condición ? e1 : e2

Si la condición es cierta
devuelve e1 y si es falsa, e2

Estructuras de control de flujo

Python	Java
if	if
if-else	if-else
x	switch
while	while
x	do-while
for	for

Condicionales

Iterativas

Sentencia condicional «if-else»

Python	Java
if condición: instrucción1 instrucción2	if (condición) { instrucción1; instrucción2; }
if condición: instrucción1 instrucción2 else: instrucción3	if (condición) { instrucción1; instrucción2; } else instrucción3;

Anidamiento de «if-else»

Python

```
if condiciónA:  
    if condiciónB:  
        instrucción1  
    else:  
        instrucción2
```

```
if condiciónA:  
    if condiciónB:  
        instrucción1  
else:  
    instrucción2
```

Java

```
if (condiciónA) {  
    if (condiciónB)  
        instrucción1;  
    else  
        instrucción2;  
}
```

```
if (condiciónA) {  
    if (condiciónB)  
        instrucción1;  
} else  
    instrucción2;
```

Llaves innecesarias
(aportan claridad)

¡Llaves necesarias!

No hay «elif»

Python	Java
if condiciónA: instrucción1 instrucción2 elif condiciónB: instrucción3 instrucción4 else: instrucción5 instrucción6	if (condiciónA) { instrucción1; instrucción2; } else if (condiciónB) { instrucción3; instrucción4; } else { instrucción5; instrucción6; }

Sentencia «do-while»

ContarDigitos.java

```
// Solo se muestran algunas instrucciones del programa principal
System.out.print("Dime un número: "); int número = entrada.nextInt();
int n = Math.abs(número);
int dígitos = 0;
do {
    dígitos++;
    n /= 10;
} while (n > 0);
System.out.println("Nº dígitos de " + número + " = " + dígitos);
```

Tiene este aspecto:

do
 sentencia
while (*condición*);

Sentencia «for»

TablaMultiplicar.java

```
// Solo se muestra el programa principal
public static void main(String[] args) {
    Scanner entrada = new Scanner(System.in);
    System.out.print("Introduce un número");
    int número = entrada.nextInt();
    System.out.println("Tabla del " + número);
    for (int i = 1; i <= 10; i++) {
        System.out.println(número + " x " + i + " = " + número * i);
    }
}
```

Tiene este aspecto:
**for (inicialización; condición; actualización)
sentencia**

La variable **i** no existe
al acabar el bucle

Métodos

MostrarAbundantes.java

```
public class MostrarAbundantes
    // Otros métodos
    private static int sumarDivisoresPropios(int n) {
        int suma = 0;
        for (int divisor = 1; divisor <= n / 2; divisor++)
            if (n % divisor == 0)
                suma += divisor;
        return suma;
    }
```

Modificador de
«método estático»

Modificador de
«visibilidad»

Métodos

MostrarAbundantes.java

```
public class MostrarAbundantes {  
    // Otros métodos  
    private static int sumarDivisoresPropios(int n) {  
        int suma = 0;  
        for (int divisor = 1; divisor <= n / 2; divisor++)  
            if (n % divisor == 0)  
                suma += divisor;  
        return suma;  
    }  
}
```

Declaración de parámetros

Tipo de retorno

Métodos

MostrarAbundantes.java

```
private static boolean esAbundante(int n) {  
    return n < sumarDivisoresPropios(n);  
}  
  
private static void mostrarAbundantesMenores(int n) {  
    System.out.print("Números abundantes menores que " + n + ": ");  
    for (int número = 1; número < n; número++)  
        if (esAbundante(número))  
            System.out.print(número + " ");  
    System.out.println()  
}
```

No devuelve nada

Sobrecarga de métodos

Maximos.java

```
private static int max(int a, int b) {  
    return a > b ? a : b;  
}  
  
private static int max(int a, int b, int c) {  
    return a > b ? (a > c ? a : c) : (b > c ? b : c);  
}  
  
private static String max(String s1, String s2) {  
    return s1.compareTo(s2) > 0 ? s1 : s2;  
}
```

Varios métodos con un mismo nombre y diferentes «firmas»

Sobrecarga de métodos

Maximos.java

```
private static int max(int a, int b) {  
    return a > b ? a : b;  
}  
  
private static int max(int a, int b, int c) {  
    return a > b ? (a > c ? a : c) : (b > c ? b : c);  
}  
  
private static String max(String s1, String s2) {  
    return s1.compareTo(s2) > 0 ? s1 : s2;  
}
```

2 parámetros
de tipo **int**

3 parámetros
de tipo **int**

2 parámetros
de tipo **String**

La clase String

LongitudCadena.java

```
// Solo se muestra el programa principal
public static void main(String[] args) {
    Scanner entrada = new Scanner(System.in);
    System.out.print("Introduce una palabra: ");
    String palabra = entrada.next();
    System.out.println("Número de caracteres: " + palabra.length());
}
```

Longitud de una cadena

Representa una
cadena «inmutable»

Concatenación de cadenas

LongitudCadena.java

```
// Solo se muestra el programa principal
public static void main(String[] args) {
    Scanner entrada = new Scanner(System.in);
    System.out.print("Introduce una palabra: ");
    String palabra = entrada.next();
    System.out.println("Número de caracteres: " + palabra.length());
}
```

The code demonstrates string concatenation. The expression `"Número de caracteres: " + palabra.length()` is highlighted with orange brackets. The word `palabra` is annotated with a bracket labeled `String`, and the method `length()` is annotated with a bracket labeled `int`.

Se convierte a String

Métodos aplicables a cadenas

Método	Equivalente Python
<code>s.length()</code>	<code>len(s)</code>
<code>s.charAt(i)</code>	<code>s[i]</code>
<code>s.substring(i, j)</code>	<code>s[i:j]</code>
<code>s1.equals(s2)</code>	<code>s1 == s2</code>
<code>!s1.equals(s2)</code>	<code>s1 != s2</code>
<code>s1.compareTo(s2) < 0</code>	<code>s1 < s2</code>
<code>s1.compareTo(s2) <= 0</code>	<code>s1 <= s2</code>
<code>s1.compareTo(s2) > 0</code>	<code>s1 > s2</code>
<code>s1.compareTo(s2) >= 0</code>	<code>s1 >= s2</code>

`0 ≤ i < s.length()`

No confundir con
`s1 == s2`

Ejemplo: contar palabras

contar_palabras.py

```
def contar_palabras(cadena):
    cantidad = 0
    anterior = ' '
    for i in range(len(cadena)):
        if cadena[i] != ' ' and anterior == ' ': # Inicio de palabra
            cantidad += 1
        anterior = cadena[i]
    return cantidad
```

Ejemplo: contar palabras

ContarPalabras.java

```
private static int contarPalabras(String cadena) {  
    int cantidad = 0;  
    char anterior = ' ';  
    for (int i = 0; i < cadena.length(); i++) {  
        if (cadena.charAt(i) != ' ' && anterior == ' ') // Inicio de palabra  
            cantidad++;  
        anterior = cadena.charAt(i);  
    }  
    return cantidad;  
}
```

Vectores

ejemplo

```
int[] viajesPorDía = new int[7];
```

Tipo «base»

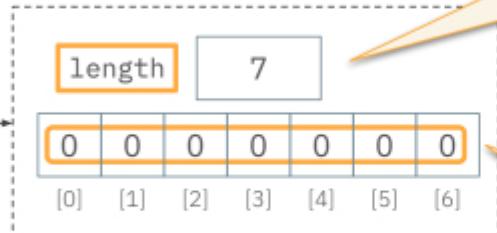
Longitud

Vectores

ejemplo

```
int[] viajesPorDía = new int[7];
```

viajesPorDía



Atributo de
«solo lectura»

Los elementos se inicializan al
«valor por defecto» del tipo base

Valor por defecto de un tipo

Tipo	Valor por defecto
int	0
double	0.0
boolean	false
char	'\u0000'
Tipo referencia	null

Declaración de vectores

ejemplo

```
int[] v1;  
int[] v2 = null;  
int[] v3 = new int[0];  
double[] precios = new double[365];  
boolean[] ocupado = new boolean[24];  
int[] alturas = {100, 200, 150, 175, 300};  
v1 = new int[]{4, -5, 2, -6, -6, 1, 2, 2};
```

Válido solo cuando creamos un vector

Ejemplo: alturas y desniveles

CarreraFondo.java

```
public static void main(String[] args) {
    // Instrucciones para crear el vector de alturas y
    // llamar al método estático calcularDesniveles
}
private static int[] calcularDesniveles(int[] alturas) {
    int[] desniveles = new int[alturas.length - 1];
    for (int i = 0; i < desniveles.length; i++)
        desniveles[i] = alturas[i + 1] - alturas[i];
    return desniveles;
}
```

[] denota el operador
de indexación

Ejemplo: alturas y desniveles

CarreraFondo.java

```
private static int cantidadTramosSubida(int[] desniveles) {  
    int tramos = 0, seguidos = 0;  
    for (int valor : desniveles)  
        if (valor > 0) seguidos++;  
        else {  
            if (seguidos > 0) tramos++;  
            seguidos = 0;  
        }  
    if (seguidos > 0) tramos++;  
    return tramos;  
}
```

Bucle for «avanzado»

Comparación de vectores

CompararVectores.java

```
public static void main(String[] args) {  
    int[] v1 = {10, 20, 30, 40, 50};  
    int[] v2 = {10, 20, 30, 40, 50};  
    System.out.println("¿v1 == v2? " + (v1 == v2));  
    System.out.println("¿v1.equals(v2)? " + v1.equals(v2));  
}
```

¡Comparan las referencias!

Comparación de vectores

CompararVectores.java

```
public static void main(String[] args) {  
    int[] v1 = {10, 20, 30, 40, 50};  
    int[] v2 = {10, 20, 30, 40, 50};  
  
    System.out.println("¿v1 == v2? " + (v1 == v2));  
    System.out.println("¿v1.equals(v2)? " + v1.equals(v2));  
  
    System.out.println("¿iguales(v1, v2)? " + iguales(v1, v2));  
}
```

Método «estático»
definido en el programa

Comparación de vectores

CompararVectores.java

```
private static boolean iguales(int[] v1, int[] v2) {  
    if (v1.length != v2.length)  
        return false;  
    else {  
        for (int i = 0; i < v1.length; i++)  
            if (v1[i] != v2[i])  
                return false;  
        return true;  
    }  
}
```

Comparación de vectores: clase Arrays

CompararVectores.java

```
public static void main(String[] args) {  
    int[] v1 = {10, 20, 30, 40, 50};  
    int[] v2 = {10, 20, 30, 40, 50};  
  
    System.out.println("¿v1 == v2? " + (v1 == v2));  
    System.out.println("¿v1.equals(v2)? " + v1.equals(v2));  
  
    System.out.println("¿iguales(v1, v2)? " + iguales(v1, v2));  
  
    boolean sonIguales = Arrays.equals(v1, v2);  
    System.out.println("¿Arrays.equals(v1, v2)? " + sonIguales);  
}
```

Método «estático» definido en la clase **Arrays**

Mostrar el contenido de un vector

MostrarVector.java

```
import java.util.Arrays;  
  
public class MostrarVector {  
    public static void main(String[] args) {  
        int[] v = {10, 20, 30, 40, 50};  
        System.out.println(v); // Muestra la referencia almacenada en v  
        System.out.println("v es " + Arrays.toString(v));  
    }  
}
```

Devuelve la cadena
"[10, 20, 30, 40, 50]"

Declaración de matrices

MatrizTraspuesta.java

```
public static void main(String[] args) {  
    // Ejemplo de programa principal (matriz con valores concretos)  
    double[][] m = {{10, 20, 30},  
                    {40, 50, 60}};  
    double[][] t = obtenerTraspuesta(m);  
  
    System.out.println("Traspuesta: " + Arrays.deepToString(t));  
}
```

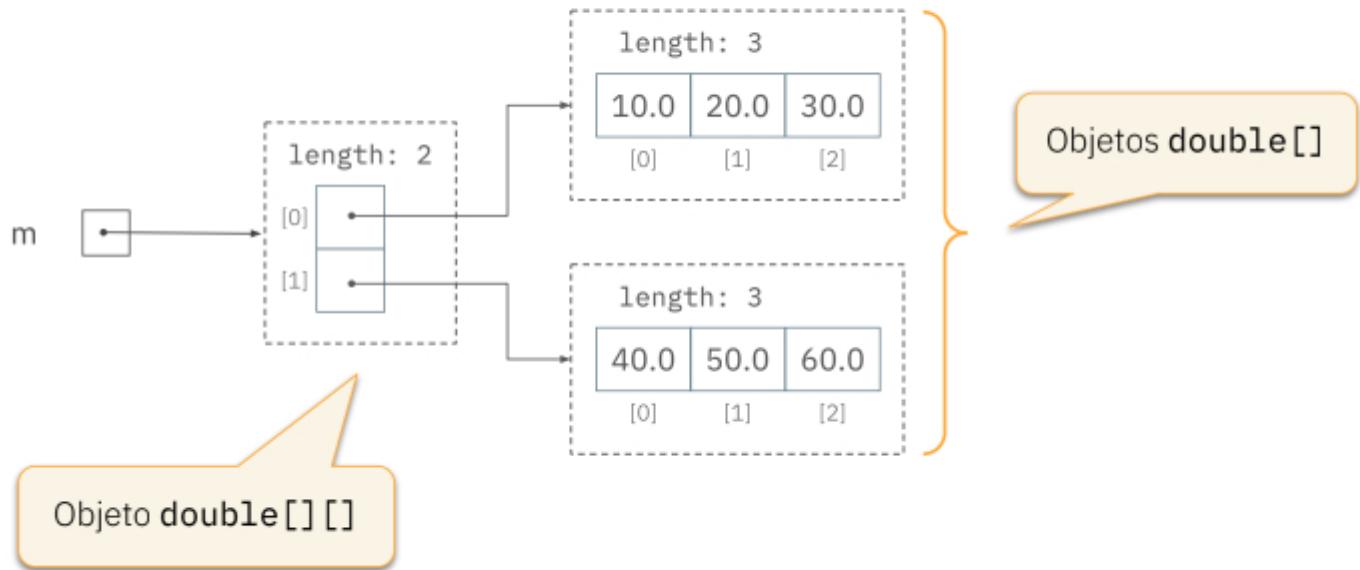
Matriz de 2 filas
y 3 columnas

Mostrar el contenido de una matriz

MatrizTraspuesta.java

```
public static void main(String[] args) {
    // Ejemplo de programa principal (matriz con valores concretos)
    double[][] m = {{10, 20, 30},
                    {40, 50, 60}};
    double[][] t = obtenerTraspuesta(m);
    System.out.println("Traspuesta: " + Arrays.deepToString(t));
}
```

Una matriz es un vector de vectores



Ejemplo: traspuesta de una matriz

MatrizTraspuesta.java

```
private static double[][] obtenerTraspuesta(double[][] matriz) {  
    // Dimensiones de la matriz traspuesta  
    int filas = matriz[0].length; // N° columnas de matriz  
    int columnas = matriz.length; // N° filas de matriz  
    double[][] traspuesta = new double[filas][columnas];  
    for (int i = 0; i < filas; i++)  
        for (int j = 0; j < columnas; j++)  
            traspuesta[i][j] = matriz[j][i];  
    return traspuesta;  
}
```

Lectura desde ficheros

ContarLineas.java

```
static int contarLineas(String nombreFichero) throws FileNotFoundException {
    int cantidad = 0; // Cantidad de lineas
    Scanner entrada = new Scanner(new File(nombreFichero));
    while (entrada.hasNextLine()) {
        String linea = entrada.nextLine();
        cantidad++;
    }
    entrada.close();
    return cantidad;
}
```

Lo estudiaremos
en el tema 2

Lectura desde ficheros

ContarLineas.java

```
static int contarLineas(String nombreFichero)
    int cantidad = 0; // Cantidad de lineas
    Scanner entrada = new Scanner(new File(nombreFichero));
    while (entrada.hasNextLine()) {
        String linea = entrada.nextLine();
        cantidad++;
    }
    entrada.close();
    return cantidad;
}
```

Creamos un objeto **File** para un fichero con ese nombre

Cerramos el objeto **Scanner** creado

Lectura desde ficheros

ContarLineas.java

```
static int contarLineas(String nombreFichero) throws FileNotFoundException {  
    int cantidad = 0; // Cantidad de lineas  
    Scanner entrada = new Scanner(new File(nombreFichero));  
    while (entrada.hasNextLine()) {  
        String linea = entrada.nextLine();  
        cantidad++;  
    }  
    entrada.close();  
    return cantidad;  
}
```

Comprobamos si quedan líneas por leer

Escritura en ficheros: clase Formatter

CrearFicheroRaices.java

```
// Crea un fichero con las raíces cuadradas de  
// los primeros n números.  
  
static void crearFicheroRaicesCuadradas(int n, String nombreFichero)  
    throws FileNotFoundException {  
  
    Formatter salida = new Formatter(new File(nombreFichero));  
  
    for (int i = 1; i <= n; i++)  
        salida.format("Raiz cuadrada de %d = %.3f\n", i, Math.sqrt(i));  
  
    salida.close();  
}
```

Creamos un objeto **Formatter** para el fichero indicado

Cerramos el objeto **Formatter** creado

Escritura en ficheros: clase Formatter

CrearFicheroRaices.java

```
// Crea un fichero con las raíces cuadradas de los números de 1 a n
static void crearFicheroRaicesCuadradas(int n, String nombreFichero)
    throws FileNotFoundException {
    Formatter salida = new Formatter(new File(nombreFichero));
    for (int i = 1; i <= n; i++)
        salida.format("Raiz cuadrada de %d = %.3f %n", i, Math.sqrt(i));
    salida.close();
}
```

Para escribir usamos **format**

Escritura en ficheros: clase Formatter

CrearFicheroRaices.java

```
// Crea un fichero con las raíces cuadradas de los números de 1 a n
static void crearFicheroRaicesCuadradas(int n, String nombreFichero)
    throws FileNotFoundException {
    Formatter salida = new Formatter(new File(nombreFichero));
    for (int i = 1; i <= n; i++)
        salida.format("Raiz cuadrada de %d = %.3f %n", i, Math.sqrt(i));
    salida.close();
}
```

%d se usa para enteros

Escritura en ficheros: clase Formatter

CrearFicheroRaices.java

```
// Crea un fichero con las raíces cuadradas de los números de 1 a n
static void crearFicheroRaicesCuadradas(int n, String nombreFichero)
    throws FileNotFoundException {
    Formatter salida = new Formatter(new File(nombreFichero));
    for (int i = 1; i <= n; i++)
        salida.format("Raiz cuadrada de %d = %.3f %n", i, Math.sqrt(i));
    salida.close();
}
```

%.3f se usa para flotantes
(3 decimales)

Escritura en ficheros: clase Formatter

CrearFicheroRaices.java

```
// Crea un fichero con las raíces cuadradas de los números de 1 a n
static void crearFicheroRaicesCuadradas(int n, String nombreFichero)
    throws FileNotFoundException {
    Formatter salida = new Formatter(new File(nombreFichero));
    for (int i = 1; i <= n; i++)
        salida.format("Raiz cuadrada de %d = %.3f %n", i, Math.sqrt(i));
    salida.close();
}
```

Salto de línea

Ejemplo: fichero de morosos

CrearFicheroMorosos.java

```
...
Scanner entrada = new Scanner(new File(nombreFicheroEntrada));
Formatter salida = new Formatter(new File(nombreFicheroSalida));
while (entrada.hasNext()) {
    String dni = entrada.next();
    double saldo = entrada.nextDouble();
    String nombre = entrada.nextLine().trim(); // Sin espacios iniciales ni finales
    if (saldo < 0) salida.format("%s %10.2f %s%n", dni, saldo, nombre);
}
salida.close();
entrada.close();
...
```

%s se usa para cadenas

