

Lenguaje Java (Resumen)

Sitio: [Centros - Cádiz](#)
Curso: Programación
Libro: Lenguaje Java (Resumen)

Imprimido por: Barroso López, Carlos
Día: martes, 21 de mayo de 2024, 23:38

Tabla de contenidos

- 1. Programa en Java. Compilación y ejecución**
- 2. Identificadores. Tipos de datos. Oper. Aritméticos**
- 3. Entrada y salida de datos**
- 4. Condicionales. Oper. de comparación y lógicos**
- 5. Bucles**
- 6. Números aleatorios: función `Math.random()`**
- 7. Arrays**
- 8. Funciones**
 - 8.1. Ejemplo
- 9. Cadenas: clase `String`**
 - 9.1. Expresiones Regulares
 - 9.2. Ejemplos
- 10. Fechas**
- 11. Enumeraciones (`enum`)**

1. Programa en Java. Compilación y ejecución

1. Estructura de un programa en Java

Programa: "Hola Mundo"

HolaMundo.java

```
/**
 * Muestra por pantalla la frase "¡Hola mundo!"
 *
 * @author J. Pozo
 */
public class HolaMundo { // Clase principal, contiene el método main (programa principal)
    public static void main(String[] args) {
        System.out.println("¡Hola mundo!"); // Salida por pantalla
    }
}
```

2. Código fuente, compilación y ejecución

Fichero fuente:

- Nombre: debe coincidir con el nombre de la clase principal.
- Extensión: .java

Ej.: *HolaMundo.java*

Compilación:

```
javac HolaMundo.java
```

Esto genera un fichero *HolaMundo.class* con el bytecode.

Ejecución:

```
java HolaMundo
```

2. Identificadores. Tipos de datos. Oper. Aritméticos

1. Identificadores

Los identificadores de variables y constantes deben ser **significativos**, es decir, deben indicar perfectamente qué información contienen.

Ejemplo:

```
x = vIp3 * Weerty - zxc; // Ilegible
```

```
precioTotal = cantidad * precio - descuento; // Legible
```

1.1. Nombre de variables: nomenclatura *lowerCamelCase*

La primera palabra se escribe en minúscula y, si se utilizan varias palabras, las siguientes empiezan con mayúscula.

Ejemplo:

```
int edadMin;
```

1.2. Constantes

Las constantes se identifican con palabras escritas en MAYÚSCULAS. Para definir una constante se debe anteponer la palabra reservada **final**.

Ejemplo:

```
final float PI = 3.1415;
```

2. Tipos primitivos

TIPO	DESCRIPCIÓN	TAMAÑO	EJEMPLO
boolean	verdadero o falso	1 bit	boolean abierto = true;
byte	número entero	8 bits	byte repeticiones = 22;
char	carácter	16 bits	char letra = 'a';
short	número entero	16 bits	short pantalones = 22;
int	número entero	32 bits	int asistentes = 22;
long	número entero	64 bits	long poblacion = 22L;
float	número con decimales	32 bits	float nota = 9.5f;
double	número con decimales	64 bits	double precio = 22.55d;

3. Inferencia de tipos

```
var identificador = <inicialización>
```

Ejemplo:

```
var mensaje = "Hola";  
System.out.println(mensaje);
```

Uso en bucles:

```
for(var i=0; i<=5; i++)  
    System.out.print(i + " ");
```

4. Operadores aritméticos

OPERADOR	NOMBRE	EJEMPLO	DESCRIPCIÓN
+	suma	20 + x	suma dos números
-	resta	a - b	resta dos números
*	multiplicación	10 * 7	multiplica dos números
/	división	altura / 2	divide dos números
%	resto (módulo)	5 % 2	resto de la división entera
++	incremento	a++	incrementa en 1 el valor de la variable
--	decremento	a--	decrementa en 1 el valor de la variable

3. Entrada y salida de datos

1. Salida por pantalla

Con salto de línea:

```
System.out.println();
```

Sin salto de línea:

```
System.out.print();
```

Ejemplo:

```
System.out.print("uno ");  
System.out.print("dos ");  
System.out.print("tres ");
```

```
uno dos tres
```

```
System.out.println("uno");  
System.out.println("dos");  
System.out.println("tres");
```

```
uno  
dos  
tres
```

2. Entrada por teclado: clase Scanner

Importar la clase Scanner:

```
import java.util.Scanner;
```

Crear un objeto de la clase Scanner:

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

Lectura de datos desde teclado:

```
s.next(); // Lee una palabra (String)
```

```
s.nextLine(); // Lee el texto introducido hasta que se pulse ENTER (String)
```

Métodos para leer un tipo primitivo:

```
s.nextInt(); // Lee un int
```

```
s.nextFloat(); // Lee un float
```

etc.

Lectura de un carácter (char):

```
s.next().charAt(0); // Lee el primer carácter de una palabra
```

Chequear el tipo del dato introducido:

```
s.hasNextInt(); // Devuelve true si el valor introducido es un int
```

```
s.hasNextFloat(); // Devuelve true si el valor introducido es un float
```

etc.

Cerrar el objeto Scanner:

```
s.close();
```

3. Ejemplos

Ejemplo 1

Programa que lee por teclado una serie de datos de una persona y los muestra por pantalla.

```
import java.util.Scanner;
public class FichaDatos
{
    public static void main(String[] args)
    {
        // Declara y crea un objeto Scanner que lee de la entrada estándar (teclado)
        Scanner sc = new Scanner(System.in);
        // Lee un texto para el nombre
        System.out.print("Nombre completo: ");
        String nombreCompleto = sc.nextLine();
        // Lee un carácter para el género
        System.out.print("Género (M/F): ");
        char genero = sc.next().charAt(0);
        // Lee un entero para la edad
        System.out.print("Edad: ");
        int edad = sc.nextInt();
        // Lee una palabra para el teléfono:
        System.out.print("Número de teléfono: ");
        String numTelefono = sc.next();

        // Muestra los datos recogidos
        System.out.println("Nombre completo: "+nombreCompleto);
        System.out.println("Género: "+genero);
        System.out.println("Edad: "+edad);
        System.out.println("Teléfono: "+numTelefono);
    }
}
```

Ejemplo 2

Programa que lee valores enteros y calcula su media.

```
import java.util.Scanner;
public class Media
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        // Inicialización de variables
        int suma = 0, contador = 0;
        // Bucle: Pide números mientras el valor introducido sea un int
        System.out.println("Introduzca números enteros: (cualquier otro carácter para terminar)");
        while (sc.hasNextInt())
        {
            int num = sc.nextInt(); // Lee el entero
            suma += num;
            contador++;
        }
        int media = suma / contador;
        System.out.println("La media es: " + media);
    }
}
```

Nota: modifica el programa anterior para calcular la media con decimales.

4. Condicionales. Oper. de comparación y lógicos

1. Sentencias condicionales

1.1. Sentencia if

```
if (condición) {
    instrucciones a ejecutar si la condición es verdadera
} else {
    instrucciones a ejecutar si la condición es falsa
}
```

1.2. Sentencia switch

```
switch(variable) {
    case valor1:
        <sentencias>
        break;
    case valor2:
        <sentencias>
        break;
    ...
    default:
        <sentencias>
}
```

2. Operadores de comparación

OPERADOR	NOMBRE	EJEMPLO	DESCRIPCIÓN
==	igual	a == b	a es igual a b
!=	distinto	a != b	a es distinto de b
<	menor que	a < b	a es menor que b
>	mayor que	a > b	a es mayor que b
<=	menor o igual que	a <= b	a es menor o igual que b
>=	mayor o igual que	a >= b	a es mayor o igual que b

3. Operadores lógicos

OPERADOR	NOMBRE	EJEMPLO	DEVUELVE TRUE CUANDO
&&	y	(7 > 2) && (2 < 4)	todas las condiciones son verdaderas
	o	(7 > 2) (2 < 4)	al menos una de las condiciones es verdadera
!	no	!(7 > 2)	la condición es falsa

5. Bucles

1. Bucle for

```
for (inicialización ; condición ; paso) {  
    <sentencias>  
}
```

2. Bucle while

```
while (expresión) {  
    <sentencias>  
}
```

3. Bucle do-while

```
do {  
    <sentencias>  
} while (expresión);
```

6. Números aleatorios: función Math.random()

La función **Math.random()** genera un número aleatorio con decimales (de tipo *double*) en el **intervalo [0, 1)**, es decir, mayor o igual que 0 y menor que 1.

1. Generar números aleatorios con decimales

Para generar un número aleatorio con decimales en el intervalo [0, N), basta con multiplicar por N el valor devuelto por la función.

```
Math.random()*N
```

Ejemplos de salida para N=10:

```
1.854993461897163
```

```
5.690351111720931
```

```
3.82310645589797
```

2. Generar números aleatorios enteros

Para generar un número aleatorio entero, basta con hacer un **casting** para convertir el número de tipo *double* que devuelve la función a tipo *int*.

```
(int)(Math.random()*N) // Devuelve un entero aleatorio entre 0 y N-1 (incluidos)
```

Ejemplos de salida para N=10:

```
0 8 0 3 8 8 7 3 2 0 8 2 1 2 9 0 6 4 5 4
```

Para que el intervalo de números aleatorios comience con un número mayor que cero, basta con sumar dicho número a la expresión anterior.

```
(int)(Math.random()*N + Inicio)
```

Ejercicio

Genera 20 números aleatorios enteros en el intervalo [50,60] (ambos incluidos)

7. Arrays

1. Definición y reserva de espacio

```
int[] n; // se define n como un array de enteros  
int n[]; // equivalente
```

```
n = new int[4]; // se reserva espacio para 4 enteros
```

O bien:

```
int[] n = new int[4];
```

A continuación, ya se pueden asignar los valores:

```
n[0] = 26;  
n[1] = -30;  
n[2] = 0;  
n[3] = 100;
```

2. Declaración e inicialización conjunta

Ejemplos:

```
int[] x = {8, 33, 200, 150, 11};
```

```
String[] color = {"rojo", "amarillo", "verde", "blanco", "azul", "negro"};
```

Nota: La reserva de memoria se realiza de forma automática.

3. Tamaño del array: propiedad "length"

```
System.out.println(color.length);
```

```
6
```

4. for por rango

```
for(tipo var : array) {...}
```

Ejemplo:

```
for (String i : color) {  
    System.out.print(i + " ");  
}
```

```
rojo amarillo verde blanco azul negro
```


8. Funciones

1. Definición

```
public static tipo nombreFuncion (parámetros) {  
    <sentencias>  
    return valor; // si el tipo de la función es distinto de void  
}
```

- "nombreFunción" debe ser **representativo** de su funcionamiento y seguir la nomenclatura *lowerCamelCase*.

2. Paso de parámetros

Muchos lenguajes de programación, como C++, permiten especificar si un parámetro se pasa por valor o por referencia. En cambio en Java no es posible.

Por defecto:

- Todos los parámetros de **tipo primitivo** se pasan siempre por valor.
- Los **arrays** se pasan siempre por referencia.

3. Ejemplo

```
/**  
 * Comprueba si un número entero positivo es primo o no.  
 * Un número es primo cuando únicamente es divisible entre  
 * él mismo y la unidad.  
 */  
 * @param x un número entero positivo  
 * @return <code>true</code> si el número es primo  
 * @return <code>false</code> en caso contrario  
 */  
public static boolean esPrimo(int x) {  
    for (int i = 2; i < x; i++) {  
        if ((x % i) == 0) {  
            return false;  
        }  
    }  
    return true;  
}
```

8.1. Ejemplo

Funciones para calcular el valor mínimo, máximo y la media de un array de enteros.

```
public class Funcion {
    public static void main(String[] args) {
        int[] numeros = {3,5,8,4,5,-7,4,9};
        System.out.println("Media: "+getMedia(numeros));
        System.out.println("Maximo: "+getMaximo(numeros));
        System.out.println("Minimo: "+getMinimo(numeros));
    }

    /* FUNCION
       @param array de enteros
       @return real media
    */
    public static float getMedia(int[]numeros){
        if (numeros.length == 0) { // Parámetro no válido
            return 0;
        }

        float media;
        int total = 0;
        for(int i=0;i<numeros.length;i++){ // Se puede usar for por rango
            total += numeros[i];
        }
        media = (float)total/numeros.length;

        return media;
    }

    /* FUNCION
       @param array de enteros
       @return entero máximo
    */
    public static int getMaximo(int[]numeros){
        if (numeros.length == 0) {
            return 0;
        }

        int maximo = numeros[0];
        for(int i = 1;i<numeros.length;i++){
            if (maximo<numeros[i]) {
                maximo = numeros[i];
            }
        }

        return maximo;
    }

    /* FUNCION
       @param array de enteros
       @return entero mínimo
    */
    public static int getMinimo(int[]numeros){
        if (numeros.length == 0) {
            return 0;
        }
        int minimo = numeros[0];
        for(int i = 1;i<numeros.length;i++){
            if (minimo>numeros[i]) {
                minimo = numeros[i];
            }
        }
        return minimo;
    }
}
```


9. Cadenas: clase String

1. Clase String

La clase String representa cadenas de caracteres. Los literales de cadenas, como "Hola Mundo", en Java se implementan como instancias de esta clase.

Estas cadenas son **constantes**, sus valores no se pueden cambiar después de su creación. Para crear cadenas de caracteres que sean modificables se debe utilizar la clase *StringBuffer*.

Sintaxis:

```
String cadena = "Hola Mundo";
```

2. Métodos

MÉTODO

Información

.length()

Devuelve el tamaño de la cadena. (int)

.charAt(int índice)

Devuelve el carácter de la posición indicada. (char)

Comparación

.equals(String str)

true si las cadenas son iguales, false en otro caso. (boolean)

.equalsIgnoreCase(String str)

Similar al anterior pero ignorando mayúsculas/minúsculas. (boolean)

.compareTo(String str)

<0 si la cadena es menor que str, 0 si son iguales, >0 si es mayor. (int)

.compareToIgnoreCase(String str)

Similar al anterior pero ignorando mayúsculas/minúsculas. (int)

Búsqueda

.indexOf(carácter/subcadena [, int posición])

Devuelve el índice donde se encuentra el carácter o la subcadena. -1 si no se encuentra. (int)

Subcadenas

.substring(int inicio [, int final])

Devuelve la subcadena comprendida entre la posición inicio y final (sin incluir). (String)

Manejo

.toLowerCase()

Devuelve la cadena convertida a minúsculas. (String)

.toUpperCase()

Devuelve la cadena convertida a mayúsculas. (String)

.trim()

Devuelve la cadena sin espacios. (String)

.replace(char oldChar, char newChar)

Devuelve la cadena reemplazando un carácter por otro. (String)

9.1. Expresiones Regulares

El método **matches(String regex)** permite saber si una cadena cumple o no una expresión regular.

1. Símbolos

Símbolo	Descripción
.	cualquier carácter
{n}	repetir n veces
{n,[m]}	repetir entre n y m veces (opcional)
+	repetir 1 o más veces
*	repetir 0 o más veces
\\	carácter de escape
[0-9][a-z][A-Z][a-zA-Z]	conjunto de caracteres
[^0-9] ...	distinto al conjunto
\\d (\\D)	dígito (no dígito)
\\s	carácter en blanco (espacio, tab, ...)
\\w	mayúscula, minúscula o _
(?i)	ignore case (ignora mayús./minús.)
	o

2. Ejemplos

Ejemplo 1: **fecha**

<code>"\\d{1,2}/\\d{1,2}/\\d{4}"</code>
<code>"11/12/2014", "1/12/2014", "11/2/2014" // true</code>
<code>"11/12/14", "11//2014", "11/12/14jose" // false</code>

Ejemplo 2: **fecha2**

<code>"\\d{1,2}/(?i)(ene feb mar abr may jun jul ago sep oct nov dic)/\\d{4}"</code>
<code>"11/dic/2014", "1/AGO/2014" // true</code>
<code>"11/abc/2014" // false</code>

9.2. Ejemplos

Ejemplo 1

```
String sCadena1 = new String("Avila");
String sCadena2 = new String("Salamanca");
String sCadena3 = new String("AVILA");
if (sCadena1.equalsIgnoreCase(sCadena2))
    System.out.println(sCadena1 + " y " + sCadena2 + " son IGUALES");
else
    System.out.println(sCadena1 + " y " + sCadena2 + " son DIFERENTES");
if (sCadena1.equalsIgnoreCase(sCadena3))
    System.out.println(sCadena1 + " y " + sCadena3 + " son IGUALES");
else
    System.out.println(sCadena1 + " y " + sCadena3 + " son DIFERENTES");
```

Ejemplo 2

```
String s = "V́ctor Cuervo";
s.substring(7);
```

Cuervo

```
String s = "En un lugar de la mancha...";
s.substring(6,11);
```

lugar

Ejemplo 3

```
String sCadena = "Esto Es Una Cadena";
System.out.println(sCadena.toLowerCase()); //esto es una cadena
System.out.println(sCadena.toUpperCase()); //ESTO ES UNA CADENA
System.out.println(sCadena.trim()); //Devuelve "EstoEsUnaCadena"
```

```
String cadena = new String("secar");
System.out.println(cadena.replace('e','a')); //sacar
```

Ejemplo 4

Versión 1

```
import java.util.Scanner;
public class ContarPalabras {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String nombre = "Hola Mundo"; // Texto donde buscar
        String busqueda; // Palabra a buscar
        System.out.println( "Dime el caracter o palabra a buscar: " );
        busqueda = sc.next();
        System.out.println("La palabra aparece " + contarPalabra(nombre,busqueda) + " veces");
    }
    public static int contarPalabra(String cadena,String palabra ){
        String cadenaM = cadena.toUpperCase();
        String palabraM = palabra.toUpperCase();

        int contador = 0;
        int numero = cadenaM.indexOf(palabraM);
        while (numero > -1) {
            contador++;
            numero = cadenaM.indexOf(palabraM, numero + 1);
        }
        return contador;
    }
}
```

Versión 2

```
public class ContadorDePalabras {
    public static void main(String[] args) {
        // Texto
        String sTexto = "palabra o palabra y palabra";
        // Texto que vamos a buscar
        String sTextoBuscado = "palabra";
        // Contador de ocurrencias
        int contador = 0;
        while (sTexto.indexOf(sTextoBuscado) > -1) {
            sTexto = sTexto.substring(sTexto.indexOf(
                sTextoBuscado)+sTextoBuscado.length(),sTexto.length());
            contador++;
        }
        System.out.println (contador);
    }
}
```

10. Fechas

Para trabajar con fechas en Java se debe incluir el paquete `java.time`:

```
import java.time.*;
```

1. Tipos

1.1. LocalDate

Representa una fecha.

Formato: **yyyy-mm-dd**

Sintaxis:

```
LocalDate unaFecha = LocalDate.of(int año, int mes, int día); // a partir de sus valores
```

```
LocalDate unaFecha = LocalDate.parse("yyyy-mm-dd"); // a partir de una cadena
```

1.2. LocalTime

Representa una hora.

Formato: **hh:mm:ss** (segundos opcionales)

Sintaxis:

```
LocalTime unaHora = LocalTime.of(int hora, int minuto, int segundo);
```

```
LocalTime unaHora = LocalTime.parse("hh:mm:ss");
```

1.3. LocalDateTime

Representa una fecha y una hora.

Formato: **yyyy-mm-ddThh:mm:ss** (segundos opcionales)

Sintaxis:

```
LocalDateTime unaFechaHora = LocalDateTime.of(int año, int mes, int día, int hora, int minuto, int segundo);
```

```
LocalDateTime unaFechaHora = LocalDateTime.parse("yyyy-mm-ddThh:mm:ss");
```

2. Fecha actual

El método **now()** devuelve la fecha y hora actual.

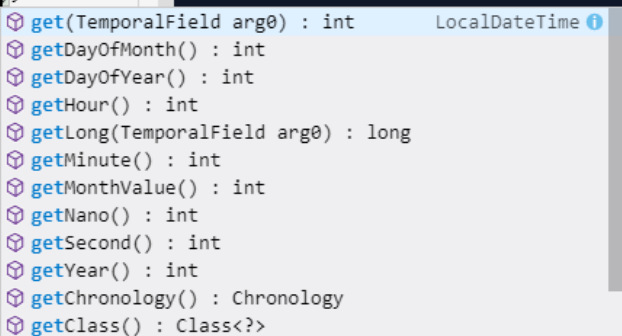
Ejemplo:

```
LocalDate.now() // devuelve la fecha actual
```

3. Partes de una fecha

Existen métodos getters para recuperar cualquier parte de una fecha.

```
5T20:40:15");  
" + hoyConHora.get();  
  
Formatter.ofPattern("dd/MM/yyyy HH:mm:ss")  
+ seisNov);
```



The screenshot shows a code editor with a snippet of Java code. A tooltip or autocomplete menu is displayed over the code, listing the methods of the `LocalDateTime` class. The methods listed are: `get(TemporalField arg0) : int`, `getDayOfMonth() : int`, `getDayOfYear() : int`, `getHour() : int`, `getLong(TemporalField arg0) : long`, `getMinute() : int`, `getMonthValue() : int`, `getNano() : int`, `getSecond() : int`, `getYear() : int`, `getChronology() : Chronology`, and `getClass() : Class<?>`.

Ejemplo:

```
unaFecha.getYear() // devuelve el año
```

4. Comparar fechas

Métodos para comparar fechas:

```
fecha1.isEqual(fecha2) // true si fecha1 y fecha2 son iguales
```

```
fecha1.isAfter(fecha2) // true si fecha1 es posterior a fecha2
```

```
fecha1.isBefore(fecha2) // true si fecha1 es anterior a fecha2
```

11. Enumeraciones (enum)

Una enumeración o tipo enumerado define una **conjunto de etiquetas**, las cuales tienen asociado un valor según su posición.

1. Definición

Sintaxis:

```
public enum NombreEnumeracion {  
    ETIQUETA1, ETIQUETA2, ...  
}
```

Nota: por convenio la lista de valores se escribe en mayúscula.

Ejemplo:

```
public enum Operacion {  
    SUMA, RESTA, MULTIPLICACION, DIVISION  
}
```

2. Variables de tipo enumerado

Declaración:

```
NombreEnumeracion variable;
```

Asignación:

```
variable = NombreEnumeracion.ETIQUETA;
```

Ejemplo:

```
Operacion op;  
op = Operacion.RESTA;
```

3. Ejemplo

```
public class PruebaEnum {  
    public static void main(String[] args) {  
        int valor1 = 10;  
        int valor2 = 20;  
        int resultado;  
        Operacion op = Operacion.RESTA;  
        switch (op) {  
            case SUMA:  
                resultado = valor1 + valor2;  
                break;  
            case RESTA:  
                resultado = valor1 - valor2;  
                break;  
            case MULTIPLICACION:  
                resultado = valor1 * valor2;  
                break;  
            case DIVISION:  
                resultado = valor1 / valor2;  
                break;  
        }  
        System.out.println("Operación: "+op+" Resultado: "+resultado);  
    }  
}
```

[Reiniciar tour para usuario en esta página](#)