LENGUAJE JAVA

MÓDULO: Programación en Java

ÍNDICE DE CONTENIDOS

1. ORIGEN DE JAVA

Surge para satisfacer las necesidades de los años 90.

La empresa **Sun Miscrosystems** pretendía crear un lenguaje que permitiese crear aplicaciones independientes del dispositivo.

Este era uno de los problemas de C/C++, los cuales dependían de la máquina

En 1991 surge **OAK** como lenguaje que pretendía permitir programar para una gran variedad de dispositivos, pero sólo se utilizó de forma interna.

1. ORIGEN DE JAVA

En 1995 **OAK** pasa a llamarse **JAVA.**

La primera versión de JAVA (1.1) era algo primitiva en muchos aspectos.

La segunda versión de JAVA (1.2) cambió tanto el lenguaje que se pasó a llamar **JAVA 2.**

Java se caracteriza por ser **portable**.

Esto se debe a que el compilador siempre obtiene el mismo código denominado **bytecode**.

El **bytecode** será interpretado por la máquina virtual (**JVM**) de Java que es independiente de cada equipo.

En 2009 Oracle compró Sun. Oracle sigue Desarrollando el lenguaje JAVA.

1.ORIGEN DE JAVA

Java 8 se lanzó en Marzo de 2014.

- Incorpora expresiones Lambda y completa la biblioteca JavaFX que mejora la concurrencia y el manejo de fechas/tiempo.
- Java 9 fue lanzado en Septiembre de 2017 -> Permite la definición de módulos.
- Java 11 se lanza en Septiembre de 2018 -> Oracle cambia la licencia de uso del JDK, a partir de ese momento no se permite el uso comercial.

- La alternativa con fines comerciales si no se quiere pagar es utilizar Oracle OpenJDK.

http://openjdk.java.net

1.ORIGEN DE JAVA

Características

- Se compila una vez el programa y, se interpreta por la máquina virtual del sistema que se use cada vez que se quiere utilizar el programa.
- Al interpretar el bytecode la máquina virtual se pueden delimitar las operaciones peligrosas o no autorizadas.
- JAVA dispone de un autenticador de bytecode por si dicho bytecode no ha sido generado por algo que no es el compilador que viene con el JDK
- Se trata de un lenguaje que
  - -Está orientado a objetos
  - -Soporta el manejo de excepciones
  - -Dispone de una amplia cantidad de bibliotecas de clases.

1.ORIGEN DE JAVA

Tipos de aplicaciones

- Aplicaciones de consola.
- Aplicaciones con interfaz gráfica de usuario (GUI).
- Applets: Aplicaciones gráficas que se ejecutan dentro de un navegador web.
- Servlets: Aplicaciones web que se ejecutan en un servidor.
- Aplicaciones para dispositivos móviles: Las aplicaciones del SSOO Android se desarrollan en JAVA, utilizando el SDK Android .

Software Development Kit

1.ORIGEN DE JAVA

Estructura del programa

- En función de su organización diremos que los programas pueden ser de dos tipos:
  - **Programas Monololíticos:** Aquellos en las que todas las operaciones del programa se realizan en una misma sección:

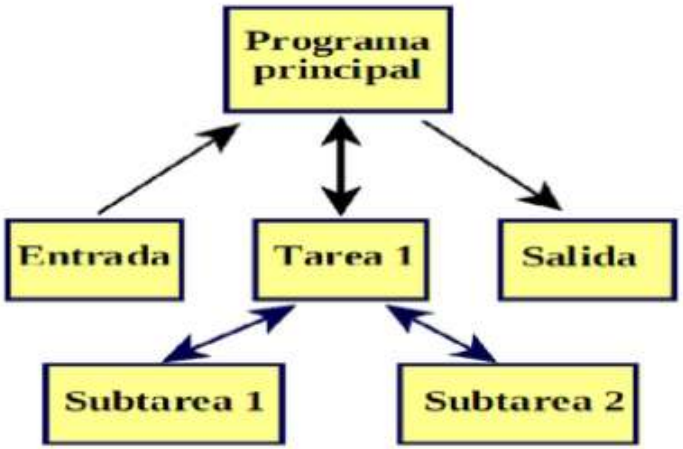
Programa monolítico

1.ORIGEN DE JAVA

Estructura del programa

- En función de su organización diremos que los programas pueden ser de dos tipos:

  - **Programas Modulares:** Aquellos en las que todas las operaciones del programa dividen en diferentes secciones. Cada una de estas secciones (**Módulos**) llevará a cabo una determinada función.



1.ORIGEN DE JAVA

Estructura del general de un programa en Java

- 1. Una sentencia de paquete (**package**)
  - Un paquete es una agrupación de clases. Podemos definirlo como una caja que contendrá todas aquellas clases que queramos utilizar en nuestro programa. Esta parte del código **No** es obligatoria.

```
1 package voleibol;
```

1.ORIGEN DE JAVA

Estructura del general de un programa en Java

- Una o varias sentencias de importación (**import**)
  - Las sentencias import nos permitirán importar clases que se encuentran en otros paquetes sin necesidad de importar todo el paquete.

```
15 import javax.swing.BorderFactory;
16 import javax.swing.JButton;
17 import java.awt.event.ActionListener;
18 import java.io.BufferedReader;
19 import java.io.FileReader;
20 import java.io.FileWriter;
21 import java.io.IOException;
22 import java.io.Writer;
23 import java.awt.event.ActionEvent;
```

1.ORIGEN DE JAVA

Definición de clases

- Las clases son unas plantillas que proporciona java para la creación de objetos, éstas pueden ser de tres tipos:
  - Públicas.
  - Privadas.
  - Protegida.

```
3 import java.sql.Connection;
10
11 public class conectar {
12     public static void main(String[]args) {
13     Connection con = null;
14
15     String username = "root";
16     String password = "testing29291";
17
18     try {
19       String url = "jdbc:mysql://localhost:3306/testing";
20       Class.forName("com.mysql.jdbc.Driver");
21       con = DriverManager.getConnection(url, username, password);
22
23       System.out.println("Connected!");
24
25       //Sentencias para la creacion de nuevos registros en la base de datos.
26       //Statement st = (Statement) con.createStatement();
27       //st.executeUpdate("INSERT INTO alumnos (idAlumnos, Nombre) VALUES ('11','hola' )");
28       //st.close();
29
30       String query = "SELECT * FROM alumnos";
```

2. ELEMENTOS DEL LENGUAJE

En un lenguaje de programación se puede definir un programa como un conjunto de sentencias, y, una sentencia como una frase o conjunto de frases informativas.

Las sentencias pueden ser de distintos tipos:

- De declaración: No implican una operación matemática o lógica pero sí que implican actividad en el ordenador.
- Ejecutables: Implican una operación matemática o lógica.
- Comentarios: Informativas, sentencias ignoradas por el computador.

Importante! **Toda sentencia ha de acabar con un punto y coma (;)**

Las sentencias se pueden considerar constituidas por tres elementos:

- Datos
- Instrucciones
- Operadores

2. ELEMENTOS DEL LENGUAJE

Para dotar al programa de funcionalidad haremos uso de los siguientes elementos:

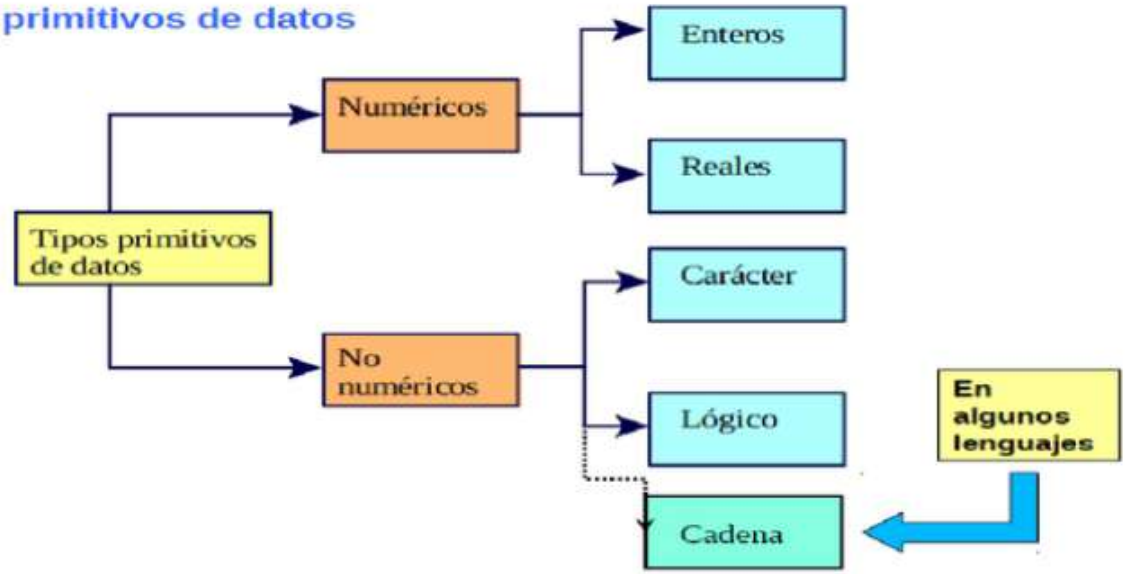
- Tipos de datos
- Variables
- Constantes
- Identificadores
- Palabras reservadas
- Operadores y expresiones
- Conversiones de tipo
- Cadenas
- Comentarios

3. TIPOS DE DATOS

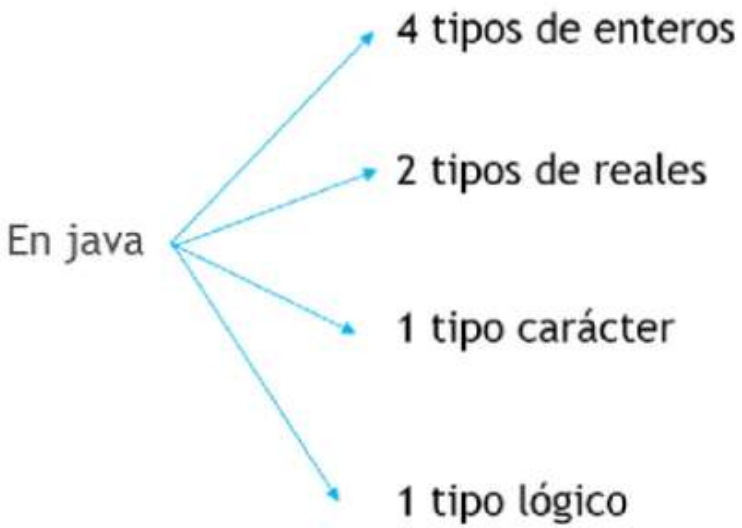
Un tipo de dato queda definido como un conjunto de valores junto con un conjunto de operaciones para manipularlo.

Los tipos de datos primitivos son aquellos tipos de datos simples, predefinidos y proporcionados por un lenguaje.

- **Tipos primitivos de datos**



3. TIPOS DE DATOS



3. TIPOS DE DATOS

- Tipos enteros (Java)

Nombre	Memoria usada (bits)	Valor mínimo	Valor máximo
`byte`	8	-128	127
`short`	16	-32768	32767
`int`	32	-2147483648	2147483647
`long`	64	$<-9.22\ 10^{18}$	$>9.22\ 10^{18}$

- Tipos reales (java)

Nombre	Memoria usada (bits)	Valor mínimo	Valor máximo
`float`	32	$\simeq 1.4\ 10^{-45}$	$\simeq 3.4\ 10^{38}$
`double`	64	$\simeq 4.9\ 10^{-324}$	$\simeq 1.7\ 10^{308}$

3. TIPOS DE DATOS

- **Tipo carácter:**
  - Se pueden utilizar dos sistemas de codificación:
    - ASCII: Permite codificar 127 o 256 caracteres
    - UNICODE: Permite codificar 65536 caracteres
  - El nombre del tipo en java es char.
  - Los caracteres están ordenados en orden alfabéticos con el código aumentando en el propio orden alfabético.
  - Un ejemplo del tipo carácter puede ser el siguiente: 'A' o 'B'.

Caracteres ASCII de control

00	NULL	(carácter nulo)
01	SOH	(inicio encabezado)
02	STX	(inicio texto)
03	ETX	(fin de texto)
04	EOT	(fin transmisión)
05	ENQ	(consulta)
06	ACK	(reconocimiento)
07	BEL	(timbre)
08	BS	(retroceso)
09	HT	(tab horizontal)
10	LF	(nueva línea)
11	VT	(tab vertical)
12	FF	(nueva página)
13	CR	(retorno de carro)
14	SO	(desplaza afuera)
15	SI	(desplaza adentro)
16	DLE	(esc.vínculo datos)
17	DC1	(control disp. 1)
18	DC2	(control disp. 2)
19	DC3	(control disp. 3)
20	DC4	(control disp. 4)
21	NAK	(conf. negativa)
22	SYN	(inactividad sincr)
23	ETB	(fin bloque trans)
24	CAN	(concolor)
25	EM	(fin del medio)
26	SUB	(sustitución)
27	ESC	(escape)
28	FS	(sep. archivos)
29	GS	(sep. grupos)
30	RS	(sep. registros)
31	US	(sep. unidades)
127	DEL	(suprimir)

Caracteres ASCII imprimibles

32	espacio	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

ASCII extendido (Página de código 437)

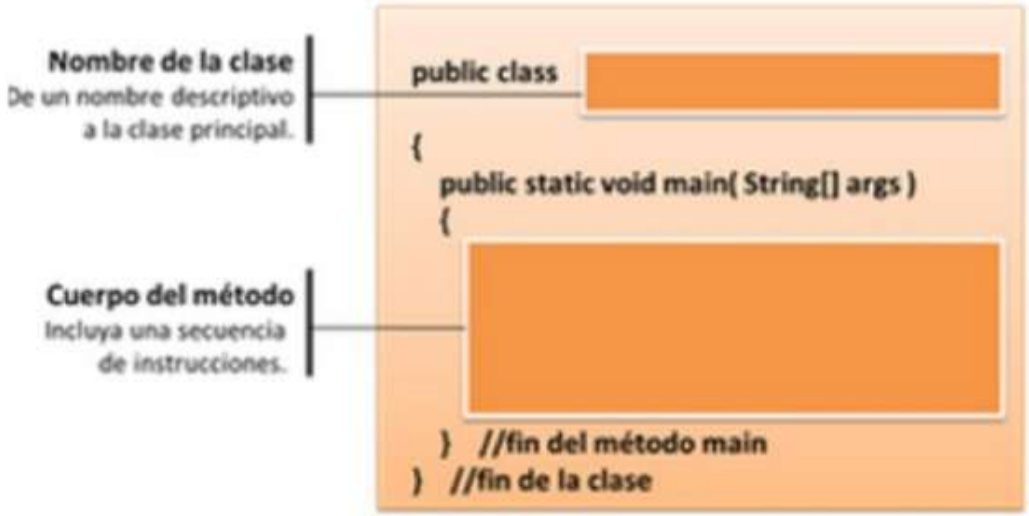
128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	▌	254	■
159	ƒ	191	┐	223	▀	255	nbsp

3. TIPOS DE DATOS

- **Tipo lógico:**
  - Este tipo primitivo se utiliza para representar los dos posibles valores lógicos: verdadero (true) o falso (false).
  - El nombre del tipo en java es boolean.
  - Los dos únicos valores posibles son true o false.

4. DECLARACIÓN DE VARIABLES

Estos elementos nos permitirán hacer que nuestro programa proporcione la funcionalidad deseada.



4. DECLARACIÓN DE VARIABLES

Centro Oficial FP Digital & Tech

Estos elementos nos permitirán hacer que nuestro programa proporcione la funcionalidad deseada.

¿Qué es String [ ] args?

```
3 public class ClasePrograma cion {
4     public static void main(String [] args) {
5
6     }
7
8 }//Fin de la clase programación
```

4. DECLARACIÓN DE VARIABLES

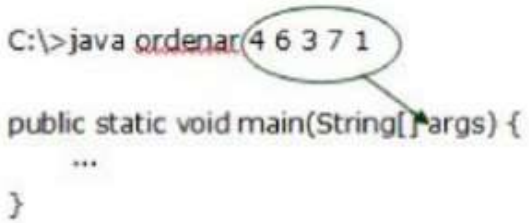
¿Qué es String [ ] args?

Es un **array** (vector) que debe aparecer de forma obligatoria como argumento del método main en un programa JAVA.

Este **array** (vector) es utilizado para mandar valores al programa en el caso de que éste no sea ejecutado desde el entorno de desarrollo (IDE).

Ejemplo: Supongamos que el programa ordenar se encarga de colocar los números que se escriben a continuación:

C:\> java ordenar 4 6 3 7 1



4. DECLARACIÓN DE VARIABLES

- El array (vector) llamado **args** contendrá todos los valores que se le manden independientemente de la tipología de estos:
    - **Números enteros.**
    - **Números en coma flotante.**
    - **Caracteres.**
    - **Cadenas.**

4. DECLARACIÓN DE VARIABLES

Variables:

- Una variable es el nombre que le asignamos a una posición o posiciones de memoria utilizadas para almacenar el valor de cierto tipo de datos.

- La sintaxis en java para declarar una variable es la siguiente:

  Tipo_de_dato nombre_de_la_variable

  Ejemplo: int numero_ruedas;

  double valor_cuenta;

- Se pueden declarar varias variables en una misma sentencia:

  Ejemplo: int numero_ruedas, numero_vasos;

- Las variables pueden inicializarse (darles un valor inicial) en la propia declaración:

  Ejemplo: int numero_ruedas = 4;

  double valor_cuenta = 3.12;

4. DECLARACIÓN DE VARIABLES

Alcance de las Variables:

- Zona del programa donde está definida.
- Estructuración de un programa en bloques. (dentro del módulo).
- En Java el bloque se indica con { }
- En Java el alcance de una variable es el bloque donde está definida.
- Si se intenta usar la variable fuera del bloque en el que se ha declarado se produce un error de variable no declarada.

5. CONSTANTES

Constantes:

- En programación una constante es una entidad parecida a una variable, pero con la diferencia de que una vez se asigna, si valor no se puede cambiar en el programa.

- La sintaxis en java para declarar una constante es la siguiente:

  final Tipo_de_dato nombre_de_la_variable

  Ejemplo: final double e = 2.7182;

```
final int LUNES = 0;
final int MARTES = 1;
final int MIERCOLES = 2;
final int JUEVES = 3;
final int VIERNES = 4;
final int SABADO = 5;
```

6. IDENTIFICADORES

Identificadores

- Los identificadores pueden construirse utilizando lo siguiente:
  - **Letras.**
  - **Dígitos.**
  - **El carácter de subrayado (_).**
  - **El signo del dólar ($).**
- Un identificador no puede comenzar con un dígito, pero puede tener cualquier longitud.
- Un identificador no puede contener caracteres:
  - Que representen en general operaciones.
  - ‘ # ´ . ? &
- Java distingue entre mayúsculas y minúsculas, por lo que la variable Datos sería distinta de la variable datos.

6. IDENTIFICADORES

6.1. PALABRAS RESERVADAS

Palabras reservadas:

- Son identificadores predefinidos que tienen un significado para el compilador y por tanto no pueden utilizarse como identificadores creados por el usuario en los programas.

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

7. TIPOS DE OPERADORES

Operadores y expresiones:

- A menudo las sentencias de programación involucran expresiones.
- Una expresión es una combinación de operadores y operandos utilizados para realizar un cálculo.
- Un operador es una entidad que representa una operación.
- En los lenguajes de programación los operadores son genéricamente los mismos y admiten una clasificación en función del tipo de operación que realizan.

- Tipos:
  - Aritméticos y de asignación
  - De incremento y decremento
  - Relacionales
  - Lógicos

7. TIPOS DE OPERADORES

Operadores aritméticos y lógicos:

- Son operadores binarios que aplican las operaciones aritméticas

Operación realizada	Operador
Suma	+
Resta	-
Multiplicación	*
División	/
Resto	%

7. TIPOS DE OPERADORES

Operadores aritméticos y lógicos:

- **Resto:**
  - **13%12=1; 12%13=12; -13%12=-1; 13%-12=1**

- **Precedencia de operadores:**
  - **Orden**

    (* , / , %) > (+, -)

  - **Ejemplos**
    - **5+4/3**
    - **5+12/5-10%3**

  - **Uso de paréntesis para alterar la precedencia**
    - **((5+12)/5)-10%3**

7. TIPOS DE OPERADORES

Operadores aritméticos y lógicos:

- Existen varios operadores que combinan una operación básica con la asignación. La idea es simplificar la operación habitual de realizar una operación sobre una variable y almacenar el resultado en esa misma variable

- Operadores:

Operador	Ejemplo	Equivalencia
+=	a+=b	a=a+b
-=	a-=b	a=a-b
*=	a*=b	a=a*b
/=	a/=b	a=a/b
%=	a%=b	a=a%b

7. TIPOS DE OPERADORES

Operadores incremento y decremento.

Son operadores que suman(incremento) o restan (decremento) una unidad a un operando entero o real.

Operadores:

Operación realizada	Operador
Incremento en una unidad	+ +
Decremento en una unidad	- -

- Ejemplos:

```
contador++;
valor--;
```

- Equivalentes a:

```
contador=contador+1;
valor=valor-1;
```

7. TIPOS DE OPERADORES

Operadores relacionales.

Expresan relaciones entre dos entidades u operandos.

Tipos:

Relación	Sintaxis
Igual	= =
Distinto	!=
Mayor	>
Menor	<
Mayor o igual	>=
Menor o igual	<=

7. TIPOS DE OPERADORES

Operadores lógicos.

Son utilizados para aceptar o rechazar proposiciones (sentencias lógicas en nuestro caso)

Tipos:

Significado	Operador
"y" lógico (AND)	&&
"o" lógico (OR)	||
"no" lógico (NOT)	!
"o" exclusivo (XOR)	^

Precedencia ! > ^ > && > ||

7. TIPOS DE OPERADORES

Operadores lógicos.

- **"o" lógico (disyunción)**

X	Y	X ˇ Y (X || Y)
V	V	V
V	F	V
F	V	V
F	F	F

- **"no" lógico (negación)**

X	¬X (!X)
V	F
F	V

7. TIPOS DE OPERADORES

Operadores lógicos.

- **"o" exclusivo (disyunción exclusiva, XOR)**

X	Y	X ˇ Y (X || Y)
V	V	F
V	F	V
F	V	V
F	F	F

8. CONVERSIONES

Tipos de conversiones

- De ensanchamiento o promoción
- De estrechamiento o contracción

8. CONVERSIONES

Tabla de Conversión de Tipos de Datos Primitivos

		Tipo destino							
		boolean	**char**	**byte**	**short**	**int**	**long**	**float**	**double**
Tipo origen	boolean	-	N	N	N	N	N	N	N
	char	N	-	C	C	CI	CI	CI	CI
	byte	N	C	-	CI	CI	CI	CI	CI
	short	N	C	C	-	CI	CI	CI	CI
	int	N	C	C	C	-	CI	CI*	CI
	long	N	C	C	C	C	-	CI*	CI*
	float	N	C	C	C	C	C	-	CI
	double	N	C	C	C	C	C	C	-

N: Conversión no permitida (un **boolean** no se puede convertir a ningún otro tipo y viceversa). **CI:** Conversión implícita o automática. Un asterisco indica que puede haber posible pérdida de datos.

C: Casting de tipos o conversión explícita.

8. CONVERSIONES

Conversiones del tipo boolean: Este tipo de datos no se puede convertir a ningún otro tipo y viceversa.

Existen tres mecanismos de conversión

- Por asignación
- Por proporción aritmética
- Por moldes (**Casting**)

8. CONVERSIONES

Conversión por asignación: Se realiza cuando un valor de un determinado tipo se asigna a una variable de otro tipo.

Ejemplo: Si dinero es una variable de tipo float y euros es una variable de tipo int -> **dinero = euros;**

Sólo se permiten conversiones de tipo **ensanchamiento**.

Conversión por proporción aritmética: Ocurre cuando se realiza una operación aritmética como la suma o la división.

Ejemplo: Si resultado es una variable de tipo float, suma es también una variable de tipo float y, contador es una variable de tipo int -> resultado = suma / contador;

La proporción aritmética es siempre de tipo ensanchamiento.

8. CONVERSIONES

Conversión por casting: Mecanismo para realizar la transformación. El molde es la instrucción que produce la conversión de tipo.

Se pueden realizar conversiones de contracción y de ensanchamiento.

En Java, el molde es un operador que se especifica como un nombre de tipo colocado entre paréntesis a la izquierda del dato a convertir. Sintaxis: (tipo) variable_a_convertir

Ejemplo:

```
Int euros;
Double dinero = 30.2;
Euros = (int) dinero;
```

8. CONVERSIONES

Conjunto de caracteres alfanuméricos.

En Java son objetos de la clase String del paquete java.Lang

Para crear una cadena hay dos formas:

```
String cadena = new String("Esto es un ejemplo");
String cadena = "Esto es un ejemplo";
```

Operaciones más comunes con cadenas:

Asignación de cadenas: **cadena1 = cadena2;**
Concatenación ("suma") de cadenas: **cadena3 = cadena1 + cadena2;**
Búsqueda (determinar si un carácter o subcadena están) dentro de la cadena.
Extracción de la subcadena de la cadena original.
Comparación de cadenas (¿son iguales o no?)

9. COMENTARIOS

Hay dos formas de realizar comentarios en un programa dependiendo de su longitud:

//Comentario de extensión una línea

En el caso de que se desee hacer un comentario de más de una línea:

/*

*Aquí podremos escribir un comentario

*Aquí también

*/

```
//Comentario de una sola línea

/*
 * Comentario multilinea
 * Esta es la forma de comentar
 * Un saludo
 */
```

9. COMENTARIOS

```
class Ejemplo {
  // Ejemplo de la estructura de un programa en Java
  public static void main(String [] args) {

    // Declaración de variables
    double total=0, suma;
    total=10.0;
    suma=5.0;

    // Operación
    total=total+suma;
    System.out.println ("total:"+ total);

  } // Fin método main
} // Fin clase Ejemplo
```

Comentarios:

1: //

2: /* */

10. IMPRIMIR POR PANTALLA

No hay sentencias de entrada y salida en el lenguaje Java. La entrada y salida se realiza utilizando bibliotecas de clases predefinidas. La mayoría de las operaciones de entrada-salida están definidas en el paquete java.io de la API Java.

Streams de lectura y escritura estándar (paquete java.Lang)

Corriente (Stream)	Propósito	Dispositivo (defecto)
`System.in`	lectura	teclado
`System.out`	escritura	monitor
`System.err`	salida de errores	monitor

10. IMPRIMIR POR PANTALLA

Para mostrar mensajes por pantalla podemos realizarlo de dos formas a través de la salida estándar:

System.out.println("mensaje"); / System.out.print("mensaje");
System.out.printf("mensaje");

```
System.out.println("How does this work?");

System.out.println("We're about to find out!");
```

11. SECUENCIAS DE ESCAPE

Secuencias de escape (uso de \):

Secuencia de escape	Significado
\t	Tabulador
\n	Línea nueva
\’	Comilla simple
\”	Comilla doble
\\	Barra invertida

11. IMPRIMIR DATOS

Escritura con el método printf()

- Para escribir por pantalla con printf(), debemos de seguir la siguiente sintaxis:

  - **%[anchura][.precisión]especificador.**
  - **Anchura: número que nos da el número mínimo de caracteres a ser impresos. Si el valor a ser impreso es más corto que este número el resultado se rellena de blancos.**
  - **Precisión: número máximo de caracteres a escribir en la salida. Para formatos en punto flotante, 'e', 'E', y 'f' la precisión es el número de dígitos tras la coma (punto) decimal.**
  - **Especificador:**

11. IMPRIMIR POR PANTALLA

Para mostrar mensajes por pantalla podemos realizarlo de dos formas a través de la salida estándar:

System.out.println(“mensaje”); / System.out.print(“mensaje”);
System.out.printf(“mensaje”);

```
System.out.println("How does this work?");

System.out.println("We're about to find out!");
```

11. IMPRIMIR POR PANTALLA

A partir de la versión 1.5 de Java con la clase Scanner y el método print/printf/println.

- Antes se realizaba con la clase BufferedReader
- Lectura con la clase Scanner (paquete java.util)
  - useLocale(Locale local)
    - Locale.US = EEUU
    - Locale.ES = España
  - next(): Lee como cadena el primer "token" que se encuentra
  - nextLine(): Lee como una cadena todo los caracteres (incluidos los espacios en blanco) hasta el fin de línea.
  - nextInt(): Lee un int.
  - nextDouble(): Lee un double.
  - nextFloat(): Lee un float.

11. IMPRIMIR POR PANTALLA

```
import java.util.*;

class ES {
  public static void main(String [] args)  {

    Scanner sc = new Scanner(System.in);
    sc.useLocale(Locale.US);

    System.out.print("Introduzca una cadena: ");
    String cad1=sc.nextLine();

    System.out.print("Introduzca una cadena sin blancos: ");
    String cad2=sc.next();

    System.out.print("Introduzca un doble: ");
    double d=sc.nextDouble();

    System.out.print("Introduzca un entero: ");
    int i=sc.nextInt();
```

11. IMPRIMIR POR PANTALLA

```
        System.out.printf(Locale.US,
        "\nLos datos tipo cadena introducidos son:"+
                          "\ncadena1: %s"+
                          "\ncadena1: %20s"+
                          "\ncadena2: %s"+
                          "\ncadena2: %10.5s\n",
                          cad1,cad1,cad2,cad2);

        System.out.printf(Locale.US,
              "\nLos datos dobles introducidos son:"+
                          "\ndoble: %7.4f"+
                          "\ndoble: %7.3f"+
                          "\ndoble: %7.2e\n", d,d,d);

        System.out.printf(Locale.US,
              "\nLos datos enteros introducidos son:"+
                          "\nentero: %d"+
                          "\nentero: %5d", i,i);

    }//Fin de main
}//Fin clase ES
```

11. IMPRIMIR POR PANTALLA

- **Salida del programa anterior**

```
Introduzca una cadena: Esto es un ejemplo
Introduzca una cadena sin blancos: Ejemplo
Introduzca un doble: 123.4567
Introduzca un entero: 123

    Los datos tipo cadena introducidos son:
    cadena1: Esto es un ejemplo
    cadena1:    Esto es un ejemplo
    cadena2: Ejemplo
    cadena2:       Ejemp

    Los datos dobles introducidos son:
    doble: 123.4567
    doble: 123.457
    doble: 1.23e+02

    Los datos enteros introducidos son:
    entero: 123
    entero:    123
```

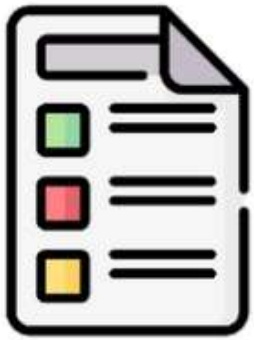
```
"\ncadena1: %s"+
"\ncadena1: %20s"+
 "\ncadena2: %s"+
  "\ncadena2: %10.5s\n"
```

```
"\ndoble: %7.4f"+
"\ndoble: %7.3f"+
 "\ndoble: %7.2e\n"
```

```
 "\nentero: %d"+
 "\nentero: %5d"
```

RESUMEN DE LA UNIDAD

- Se han analizado los orígenes de Java.
- Las variables son estructuras que permiten almacenar valores, de distinto tipo de datos.
- Las constantes nos permiten almacenar valores que no cambian.
- Se pueden agregar anotaciones al código que nos permiten realizar aclaraciones.



BIBLIOGRAFÍA/WEBGRAFÍA

- Thierry Richard. “Los fundamentos del lenguaje Java”.
- Pressma, Roger. “Ingeniería del software”.
- Vozmediano,A.M. “Java para novatos”.
- Gervás, Luc. “Aprender los fundamentos del lenguaje Java”

RECURSOS DE INTERÉS

- **Tipos de datos en Java:** https://acesse.dev/6LYTe
- **¿Qué es Java?:** https://encr.pw/2uo0g
- **Conversiones en Java:** https://l1nq.com/6ktu4
- **Comentarios en Java:** https://l1nq.com/ivNxG