

Índice

Java SE y JGrasp

Objetivos

- 1) Instalar Java SE
- 2) Insertar caracteres en un documento
- 3) Conocer los distintos paradigmas de programación
- 4) Instalar y configurar el IDE de JGrasp
- 5) Crear una aplicación en JGrasp

Elementos del lenguaje

Objetivos

- 1) Saber definir identificadores
- 2) Conocer las palabras reservadas
- 3) Escribir comentarios en un programa

Tipos de datos

Objetivos

- 1) Tipos primitivos numéricos: identificación y valores
- 2) Tipo primitivo booleano: identificación y valores
- 3) Tipo primitivo de carácter: identificación y valores
- 4) Tipo de cadena: identificación y valores

Constantes y variables

Objetivos

- 1) Saber qué es una constante
- 2) Saber qué es una variable
- 3) Saber declarar e inicializar constantes y variables de cada uno de los tipos de datos.

Salida y entrada de datos

Objetivos

- 1) Mostrar datos por consola saltando de línea
- 2) Mostrar datos por consola sin saltar de línea
- 3) Mostrar datos con formato
- 4) Lectura de datos
- 5) Detener control, dormir y limpiar la consola

Asignación de variables y algoritmos

Objetivos

- 1) Asignación de variables del mismo tipo
- 2) Algoritmos
- 3) Conversión de tipos primitivos numérico

Expresiones y operadores

Objetivos

- 1) Concatenar cadenas
- 2) Expresiones aritméticas. Truncar y redondear
- 3) Comparaciones numéricas y de cadenas
- 4) Expresiones lógicas. Validación de datos de entrada.
- 5) Operadores de bits
- 6) Operadores de acumulación y condicional
- 7) Tabla de orden de prioridad entre operadores. Operador !
- 8) La librería Math: Números aleatorios

Empaquetar y distribuir

Objetivos

- 1) Empaquetar una aplicación o librería con el JDK
- 2) Descargarse e instalar aplicación launch4j
- 3) Generar una ejecutable "pinchable"

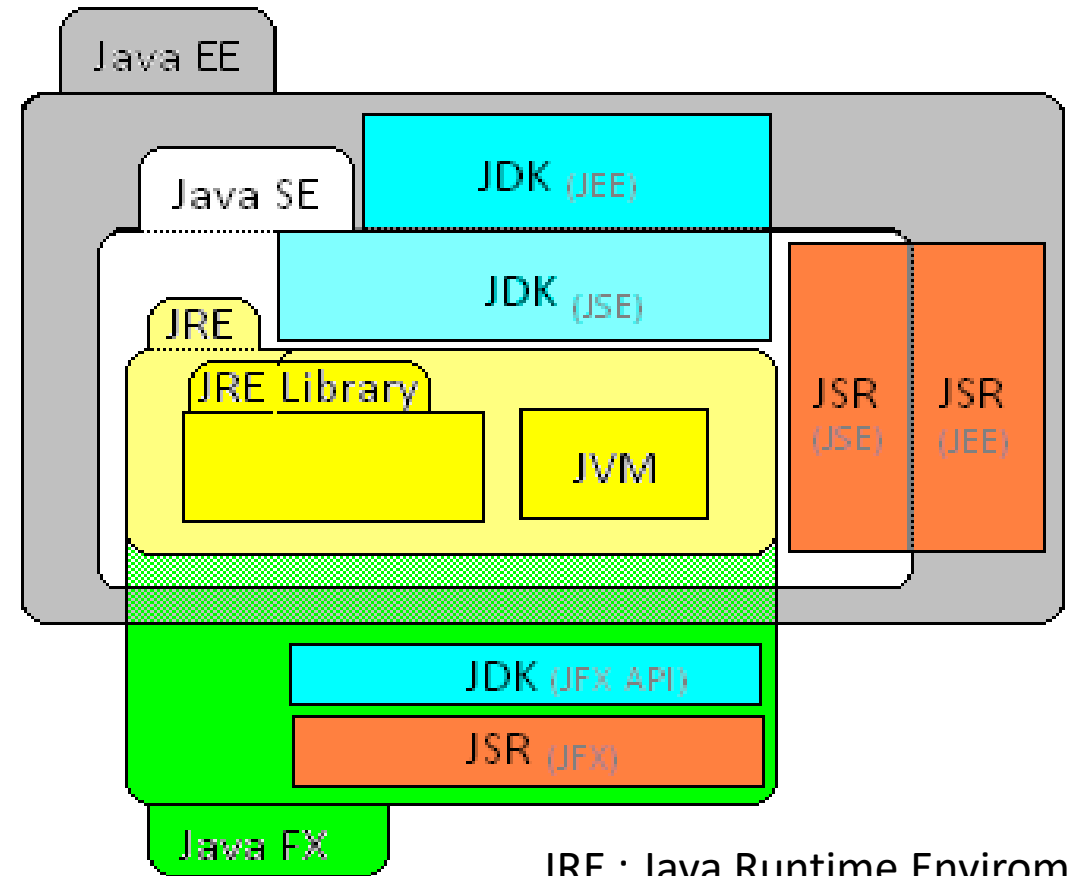
Java SE y JGrasp

Objetivos

- 1) Instalar Java SE
- 2) Insertar caracteres en un documento
- 3) Conocer los distintos paradigmas de programación
- 4) Instalar y configurar el IDE de JGrasp
- 5) Crear una aplicación en JGrap

Plataforma de java

La plataforma Java es el nombre de un entorno o plataforma de computación originaria de [Sun Microsystems](#), capaz de ejecutar [aplicaciones](#) desarrolladas usando el [lenguaje de programación Java](#) u otros lenguajes que compilen a [bytecode](#) y un conjunto de [herramientas de desarrollo](#). En este caso, la plataforma no es un [hardware](#) específico o un [sistema operativo](#), sino más bien una [máquina virtual](#) encargada de la ejecución de las aplicaciones, y un conjunto de [bibliotecas](#) estándar que ofrecen una funcionalidad común.



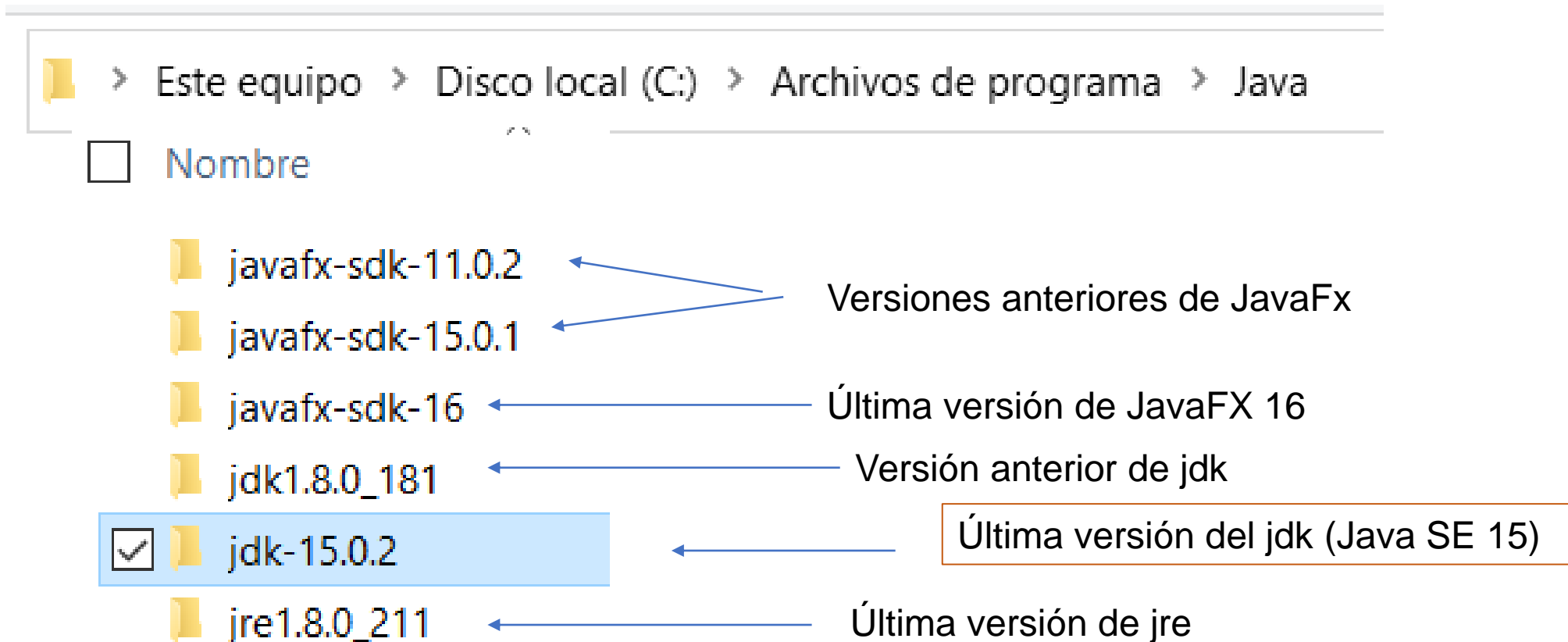
JRE : Java Runtime Enviroment
JDK: Java Development Kit
JEE: Java Enterprise Edition
JSR: Java Specification Request
JVM: Java Virtual Machine

Versiones de Java Standard Edition (Java SE)

VERSION	FECHA	VERSION	FECHA
Java 7	28 de julio de 2011		
Java 8	18 de marzo de 2014		
Java SE 9	21 de septiembre de 2017		
Java SE 10	20 de marzo de 2018		
Java SE 11	25 de septiembre de 2018		
Java SE 12	19 de marzo de 2019		
Java SE 13	16 de septiembre de 2019		
Java SE 14	19 de marzo de 2020		
Java SE 15	15 de septiembre de 2020		
Java SE 16	16 de marzo de 2021		

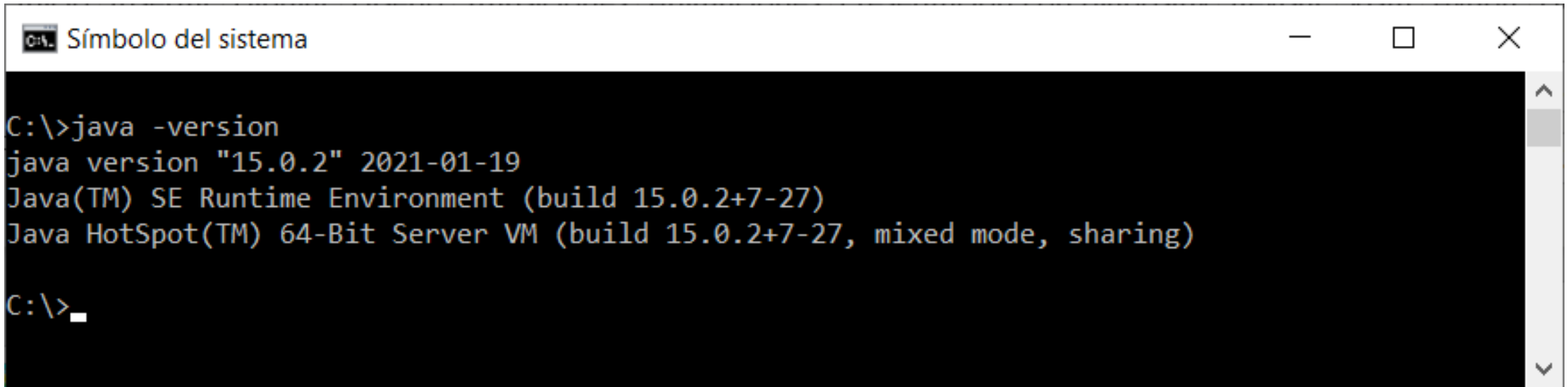
Comprobar si está instalado el JDK

1) Ver si en el subdirectorio “Java” de “Archivos de programa” en “C” hay alguna carpeta con el nombre de jdk y la versión



Comprobar si está instalado el JDK (II)

2) Ejecutar desde la consola del sistema operativo el comando “java -version”

A screenshot of a Windows Command Prompt window. The title bar at the top reads 'Símbolo del sistema' and includes standard window controls (minimize, maximize, close). The command prompt shows the command 'C:\>java -version' being executed. The output is displayed in three lines: 'java version "15.0.2" 2021-01-19', 'Java(TM) SE Runtime Environment (build 15.0.2+7-27)', and 'Java HotSpot(TM) 64-Bit Server VM (build 15.0.2+7-27, mixed mode, sharing)'. The prompt 'C:\>' is shown again at the bottom, followed by a cursor. A vertical scrollbar is visible on the right side of the command prompt area.

```
C:\>java -version
java version "15.0.2" 2021-01-19
Java(TM) SE Runtime Environment (build 15.0.2+7-27)
Java HotSpot(TM) 64-Bit Server VM (build 15.0.2+7-27, mixed mode, sharing)

C:\>_
```

Instalar Java SE

The screenshot shows the Oracle Java SE Downloads page. The browser's address bar displays the URL `oracle.com/es/java/technologies/javase-downloads.html`. The page header includes the Oracle logo and navigation links: `Productos`, `Recursos`, `Soporte`, `Eventos`, and `Desarrollador`. Below the header, the breadcrumb trail reads `Java / Technical Details / Java SE / Java SE Downloads`, and a button for `Java SE Subscriptions` is visible on the right. The main heading is `Java SE Downloads`, followed by the subtitle `Java Platform, Standard Edition`. The section `Java SE 16` is highlighted, with the text `Java SE 16.0.1 is the latest release for the Java SE Platform`. A list of links on the left includes `Documentation`, `Installation Instructions`, `Release Notes`, and `Oracle License`. On the right, under the heading `Oracle JDK`, there are two download links: `JDK Download` and `Documentation Download`. Three blue arrows point from text boxes to these elements: one from the top right to the address bar, one from the middle right to the `JDK Download` link, and one from the bottom right to the `Documentation Download` link. The bottom of the image shows a taskbar with icons for `eclipse.png`, `geany.svg`, `jgrasp.png`, and `jcreator.png`, along with a `Mostrar todo` button.

Oracle

Productos Recursos Soporte Eventos Desarrollador

Java / Technical Details / Java SE / Java SE Downloads

Java SE Subscriptions

Java SE Downloads

Java Platform, Standard Edition

Java SE 16

Java SE 16.0.1 is the latest release for the Java SE Platform

- Documentation
- Installation Instructions
- Release Notes
- Oracle License

Oracle JDK

JDK Download

Documentation Download

Para obtener dicha página escribir en un buscador "Java SE" y pulsar sobre el enlace de "Oracle"

Pulsar para descargar JDK

Pulsar para descargar la ayuda

eclipse.png geany.svg jgrasp.png jcreator.png

Mostrar todo

Entorno de Desarrollo Integrado

Un IDE, o también llamado **Entorno de Desarrollo Integrado**, es un programa que nos ayuda en la tarea de programar de modo que facilita enormemente el proceso de desarrollo y depuración de un software. Para ello cuenta con una serie de herramientas como el editor, compilador, consola y depurador e incluso funciones de autocompletado de código y resaltado de sintaxis inteligente.

Entornos de desarrollo integrados para java:

1. Eclipse
2. Netbeans
3. IntelliJ IDEA
4. Codenvy
5. JCreator
6. jGRASP
7. Geany



Instalación y configuración de JGrasp

The image shows a screenshot of the jGRASP website with several annotations in orange boxes and blue arrows indicating the installation process:

- Página web**: Points to the browser address bar showing `jgrasp.org`.
- Descarga**: Points to the [Download](#) link in the left sidebar.
- Instalación**: Points to the [Installation](#) link in the **Tutorials (PDF)** list.

The website content includes the **jGRASP** logo, the tagline "An Integrated Dev", and a navigation menu with links: Home, Download, Contact Us, Team Members, Resources, Archive, and Privacy Policy. The main text states: "The current jGRASP releases are version 2.0.6_07" and "Android Studio 3.6.2 is compatible with jGRASP".

A large blue arrow points from the [Download jGRASP](#) link to a section detailing the download options:

jGRASP 2.0.6_07 (November 25, 2020) - requires Java 8 or higher

- jGRASP exe** Windows (Vista or Higher): self-extracting executable (6,532,840 bytes).
- jGRASP pkg** macOS (High Sierra or Higher): pkg install file (requires admin access to install) (7,749,184 bytes).
- jGRASP zip** Linux, UNIX, and other systems: zip file (7,774,907 bytes).

Tablas de caracteres

<https://elcodigoascii.com.ar/>

Para representar los caracteres se tienen las tablas de caracteres

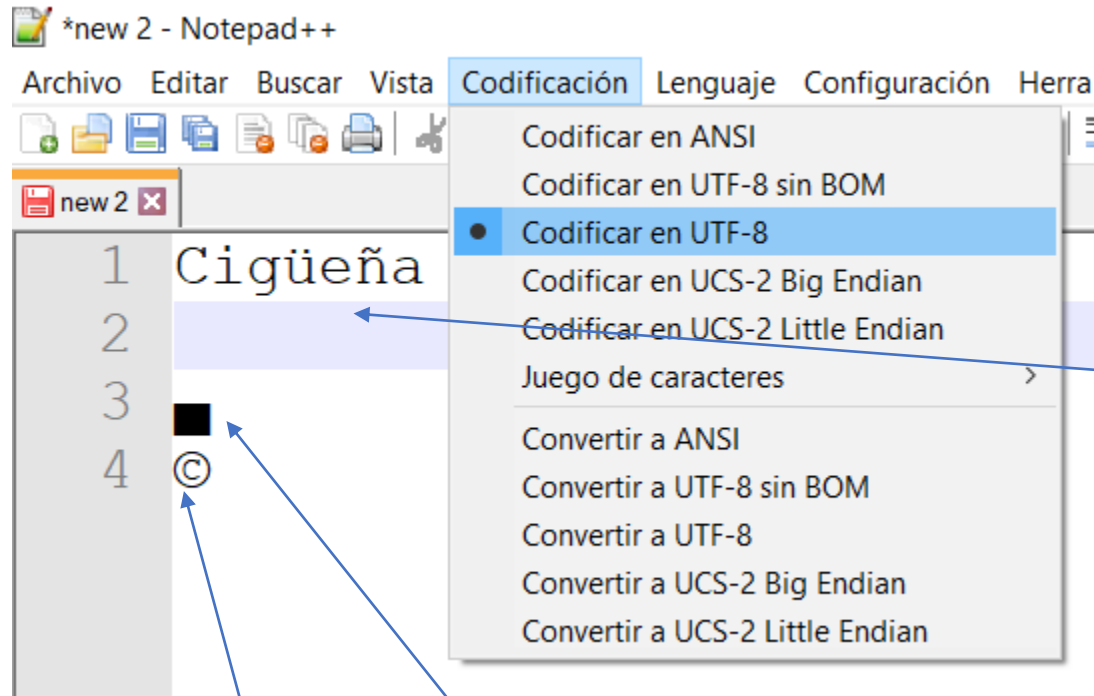
La tabla **ASCII** permite representar un total de 256 caracteres.

La tabla **UTF-8** amplia la tabla ASCII para representar un total de 65536 caracteres.

La tabla **ISO-8859-1** define los caracteres latinos

[illegible]

Insertar caracteres en un documento



1. En el documento se elige el tipo del conjunto de caracteres a utilizar

2. Se pueden introducir caracteres pulsando las teclas del teclado.

3. Para introducir caracteres ASCII que no aparecen en el teclado se hace lo siguiente:

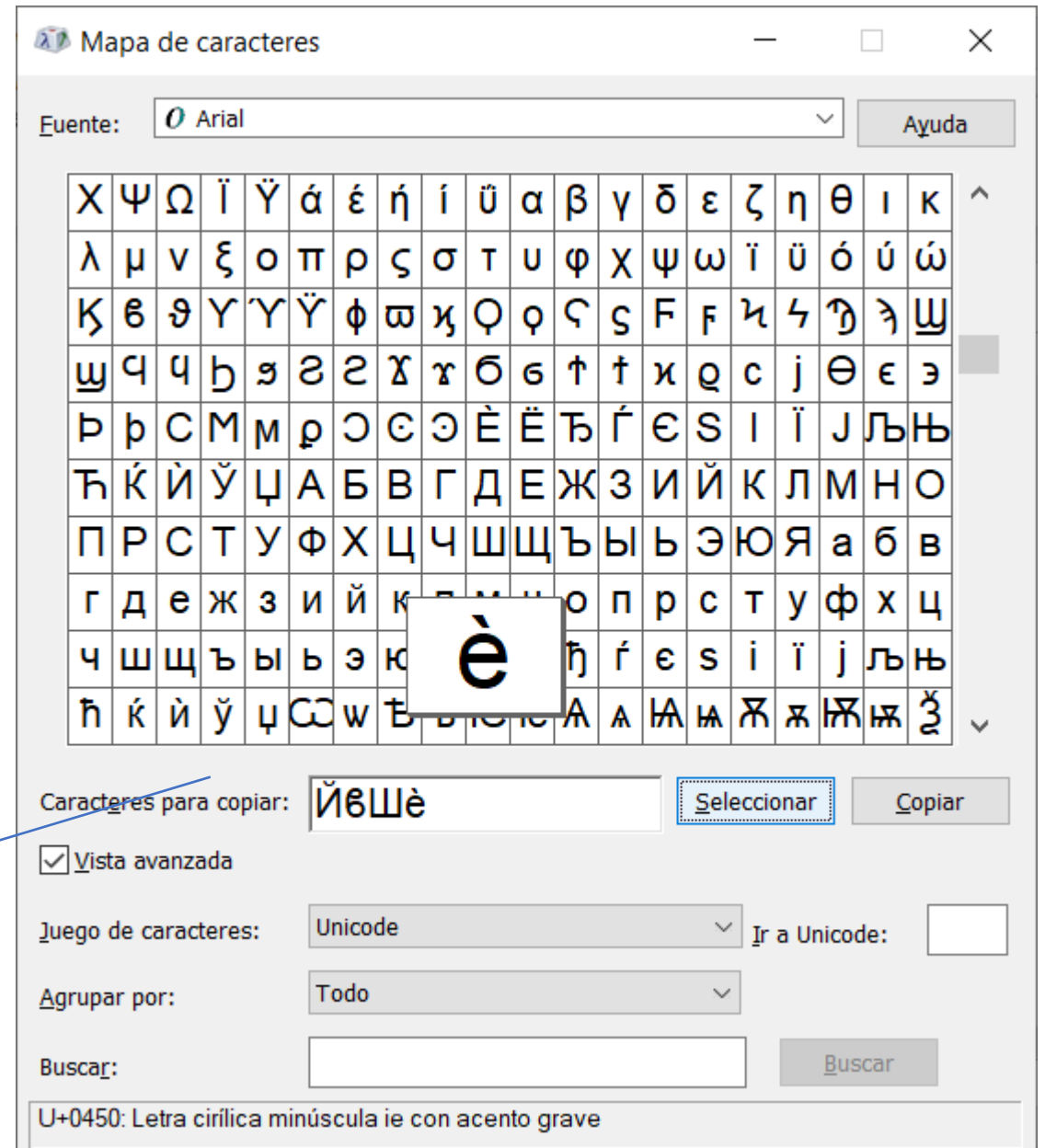
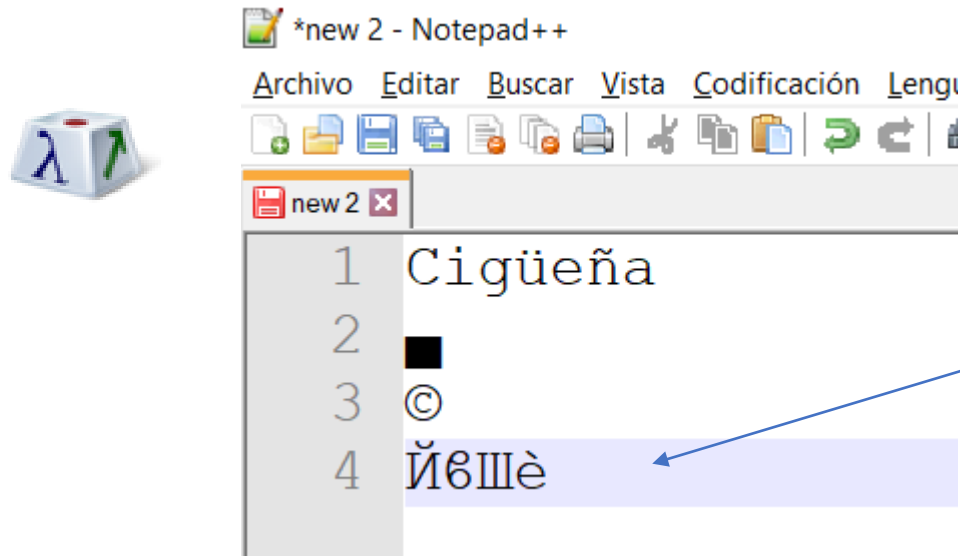
- Presiona la tecla “Alt” sin soltarla
- Presiona con el teclado numérico el código del carácter ASCII
- Deja de presionar la tecla “Alt”

Alt + 184

Alt + 220

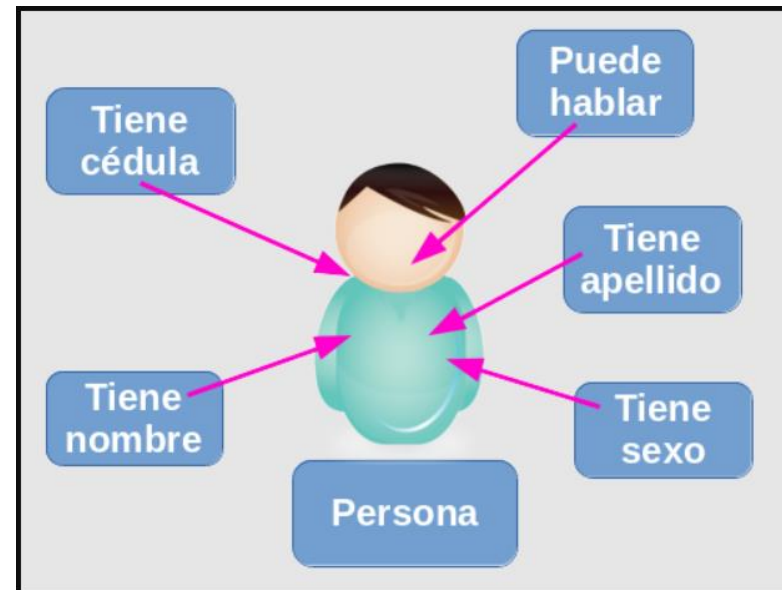
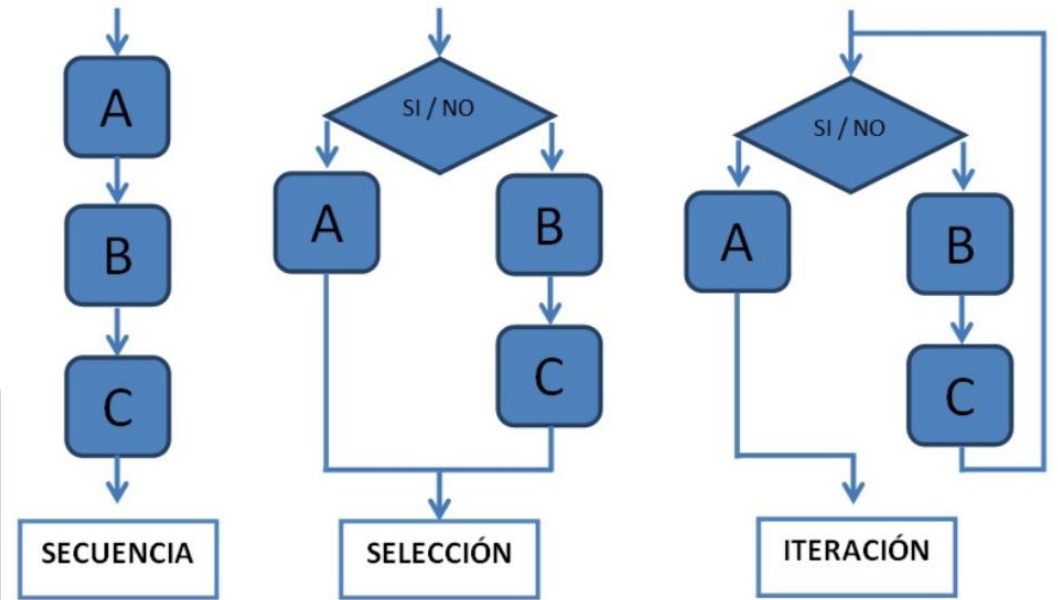
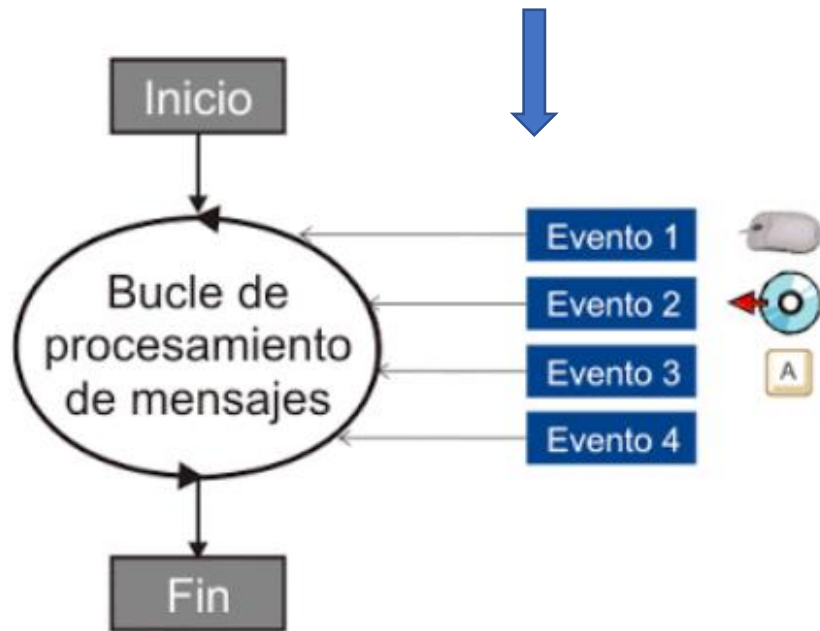
Insertar caracteres en un documento (II)

Se puede utilizar en Windows la aplicación de “Mapa de caracteres” para poder escribir caracteres de distintos conjuntos de caracteres en un tipo de fuente.

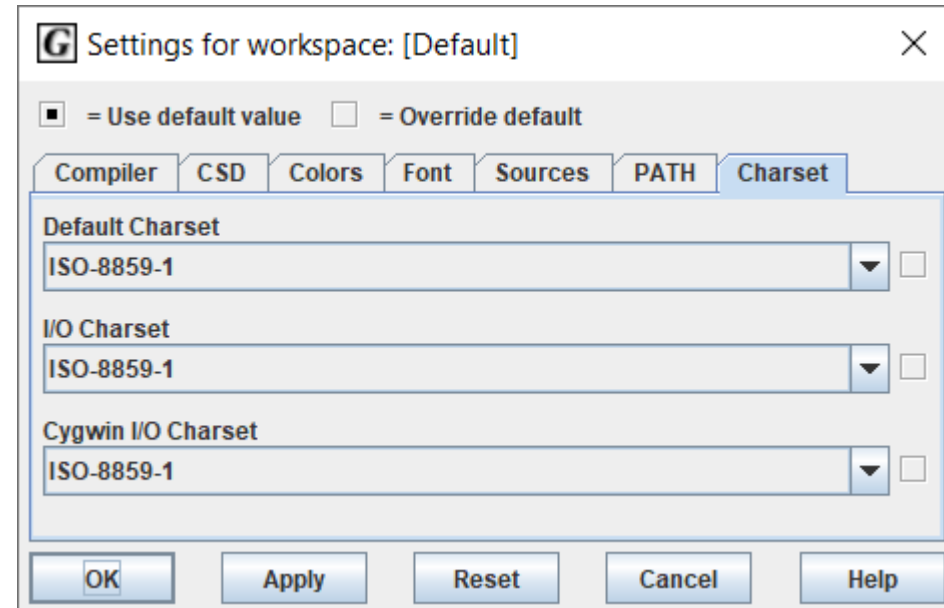
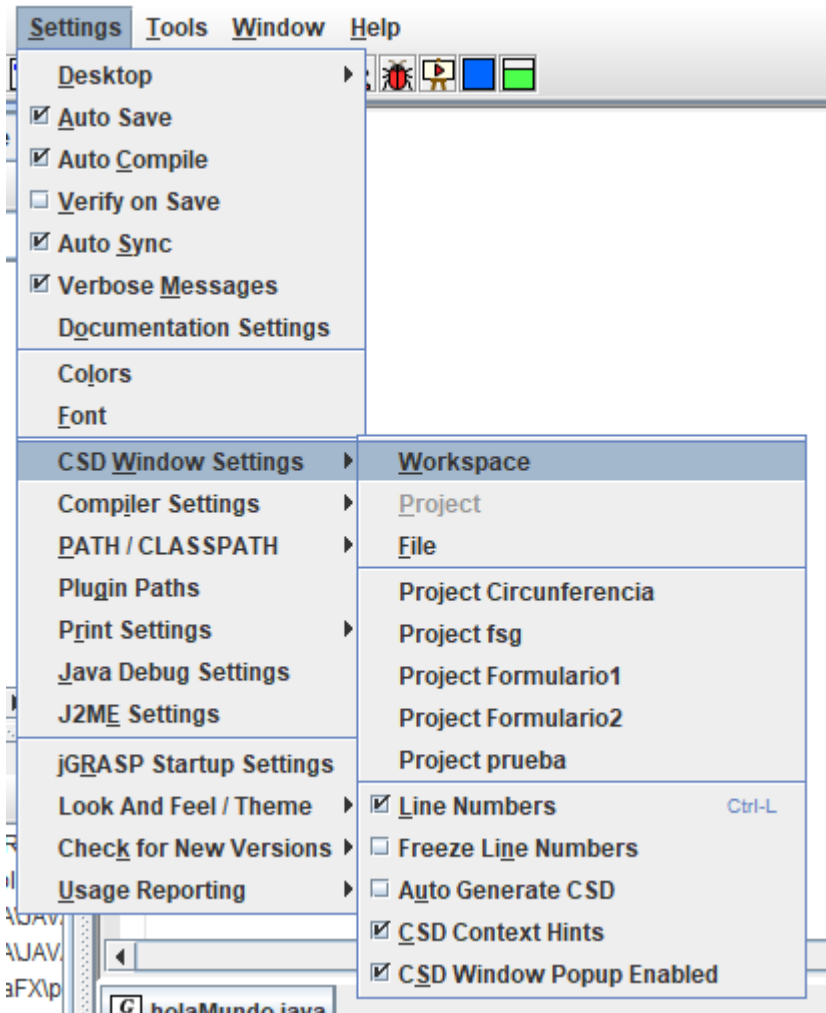


Paradigmas de programación

1. Programación estructurada
2. Programación orientada a objetos
3. Programación orientada a eventos

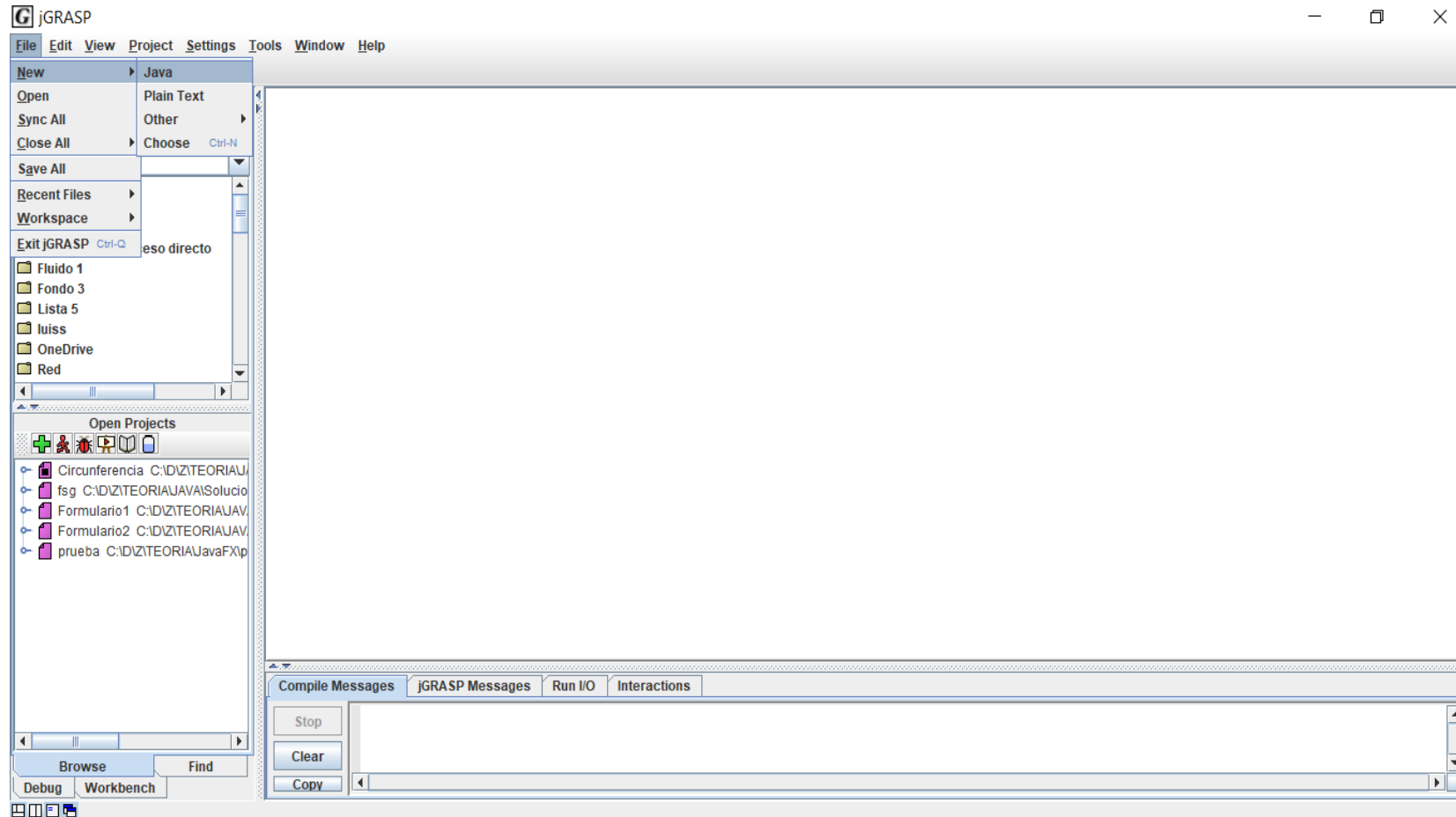


Instalación y configuración de JGrasp (II)



Se selecciona el conjunto de caracteres "ISO-8859-1" en cada uno de los apartados de la pestaña "Charset"

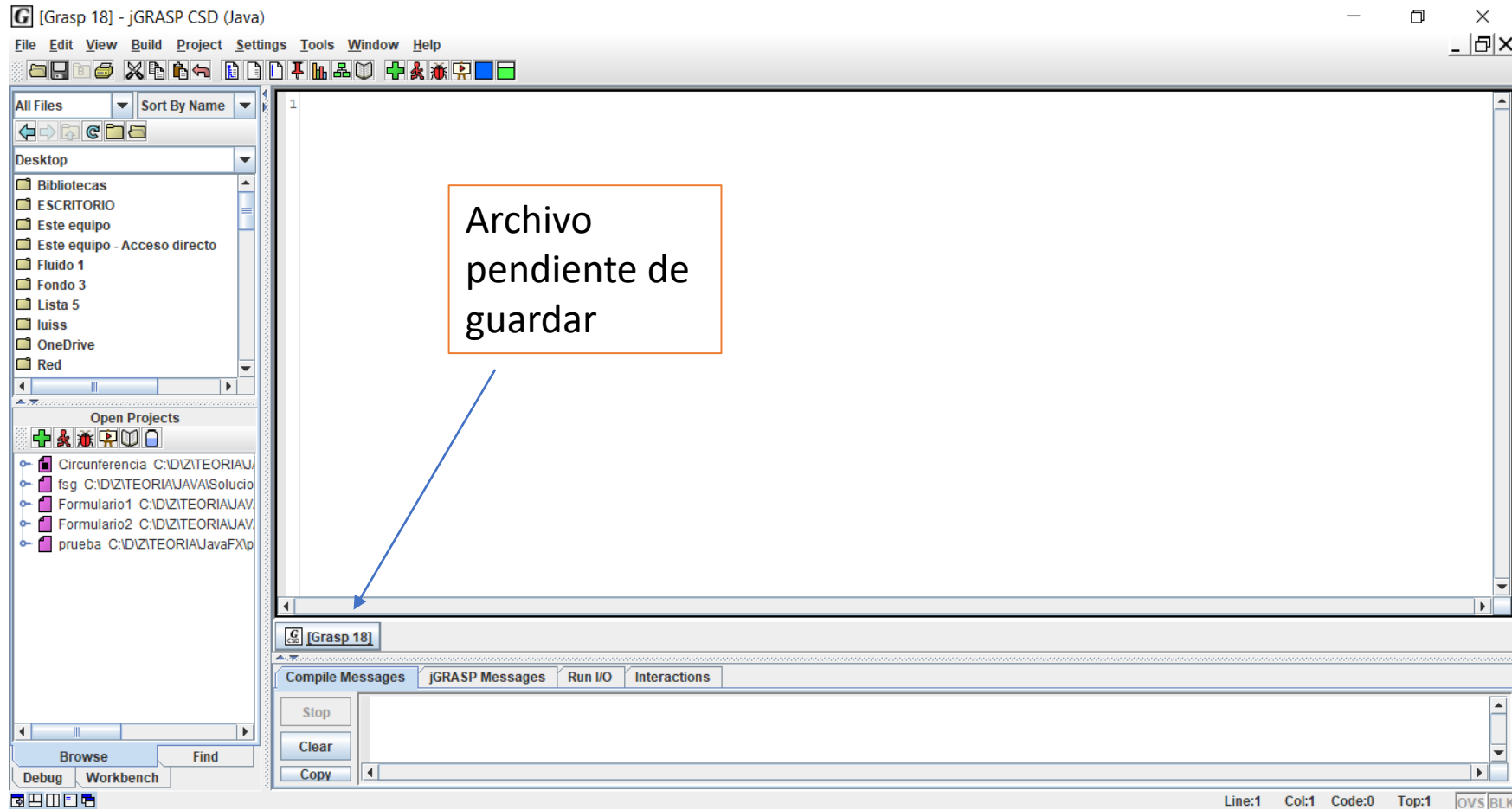
Crear una aplicación Java en JGrasp



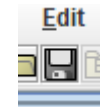
Pulsar sobre la pestaña “File”, seleccionar “New” y pulsar sobre “Java”



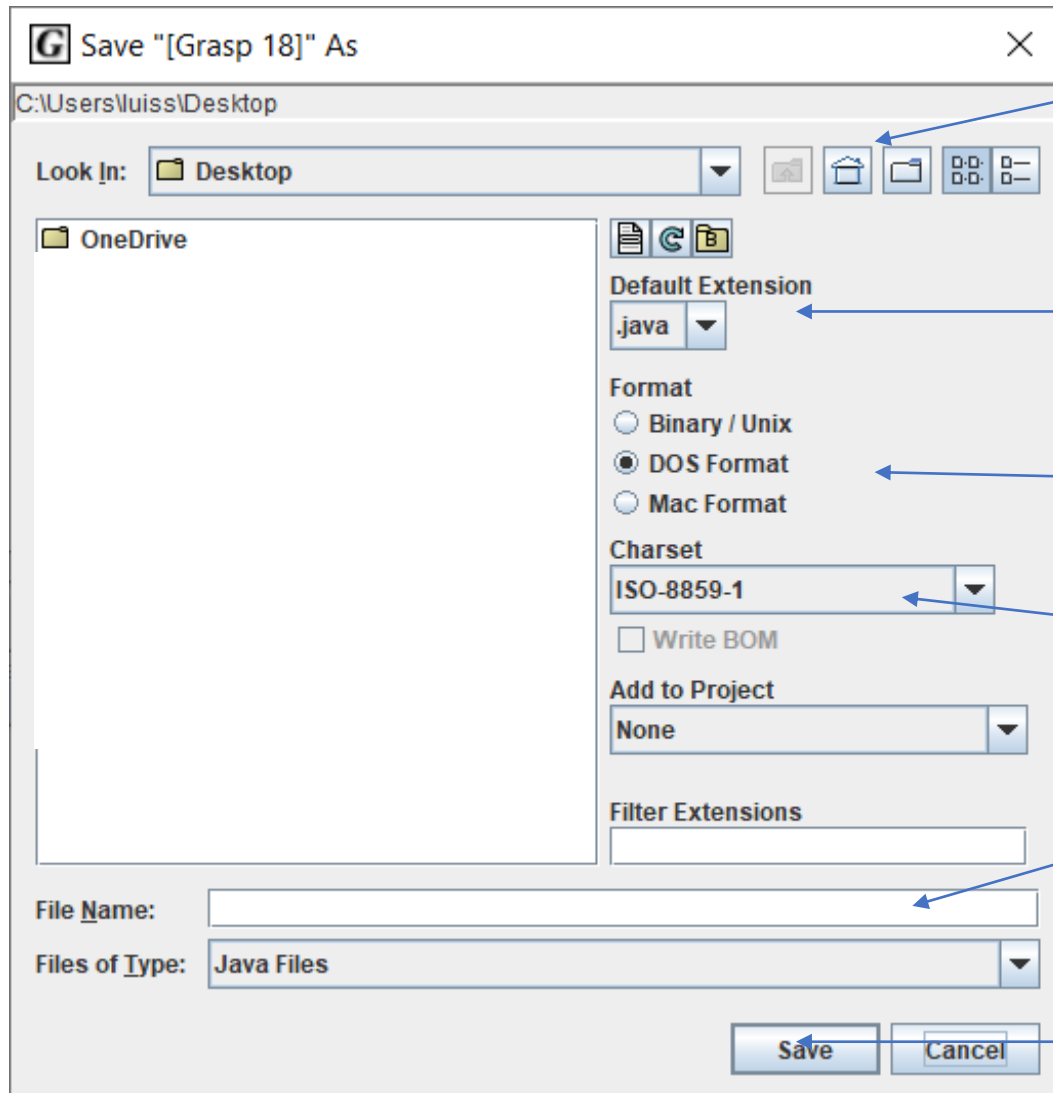
Crear una aplicación Java en JGrasp (II)




Pulsar sobre el icono



Crear una aplicación Java en JGrasp (III)



Moverse por el directorio y crear una carpeta donde se almacenarán todas las aplicaciones del tema  Practicas Tema 01


Seleccionar Java (valor por defecto)


Seleccionar como formato DOS Format (valor por defecto)

Seleccionar conjunto de caracteres ISO-8859-1

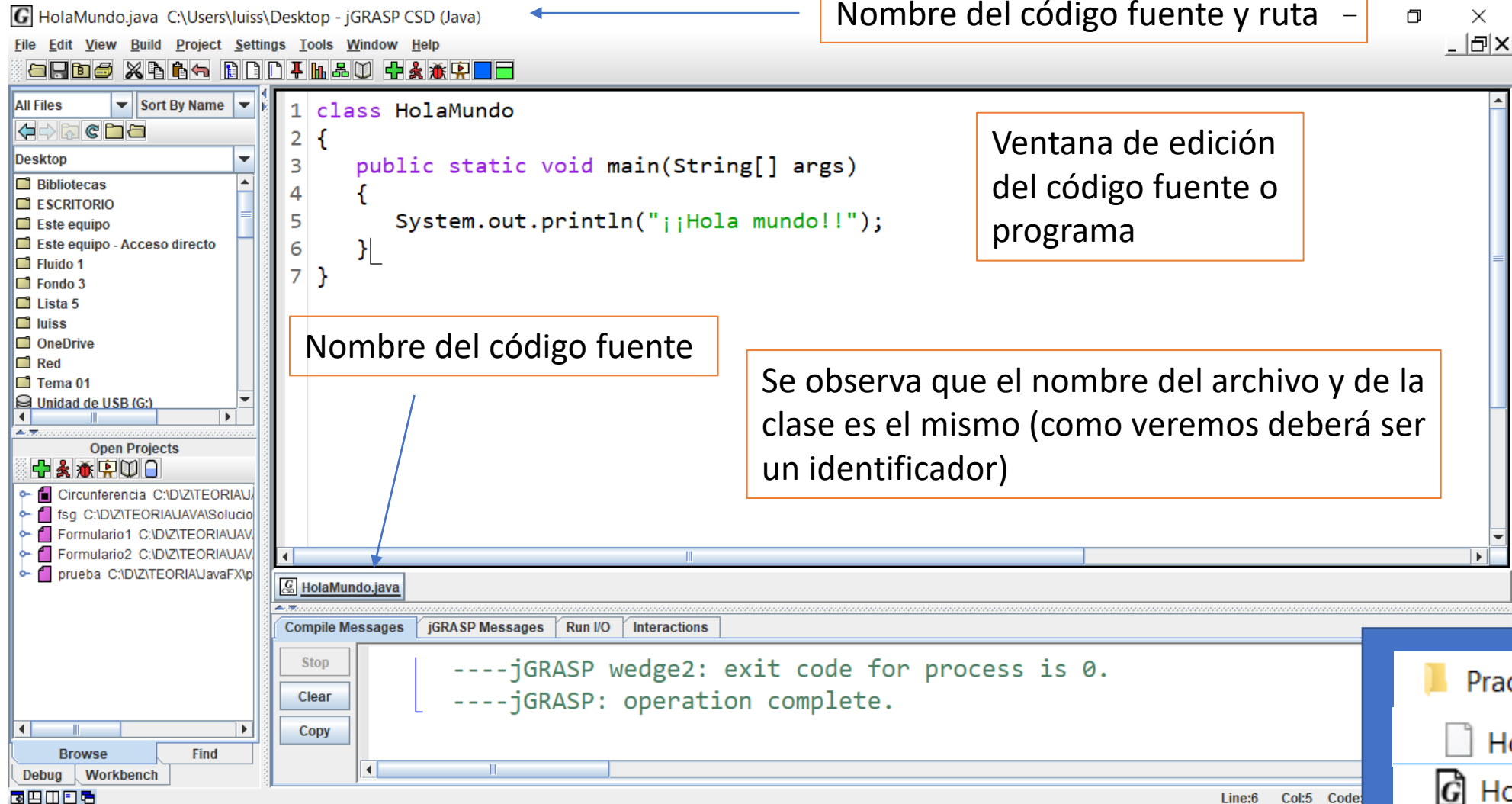
Escribir nombre del archivo fuente (por ejemplo **HolaMundo**)

Pulsar salvar

 Practicas Tema 01

 HolaMundo.java

Crear una aplicación Java en JGrasp (IV)



Nombre del código fuente y ruta

Ventana de edición
del código fuente o
programa

Nombre del código fuente

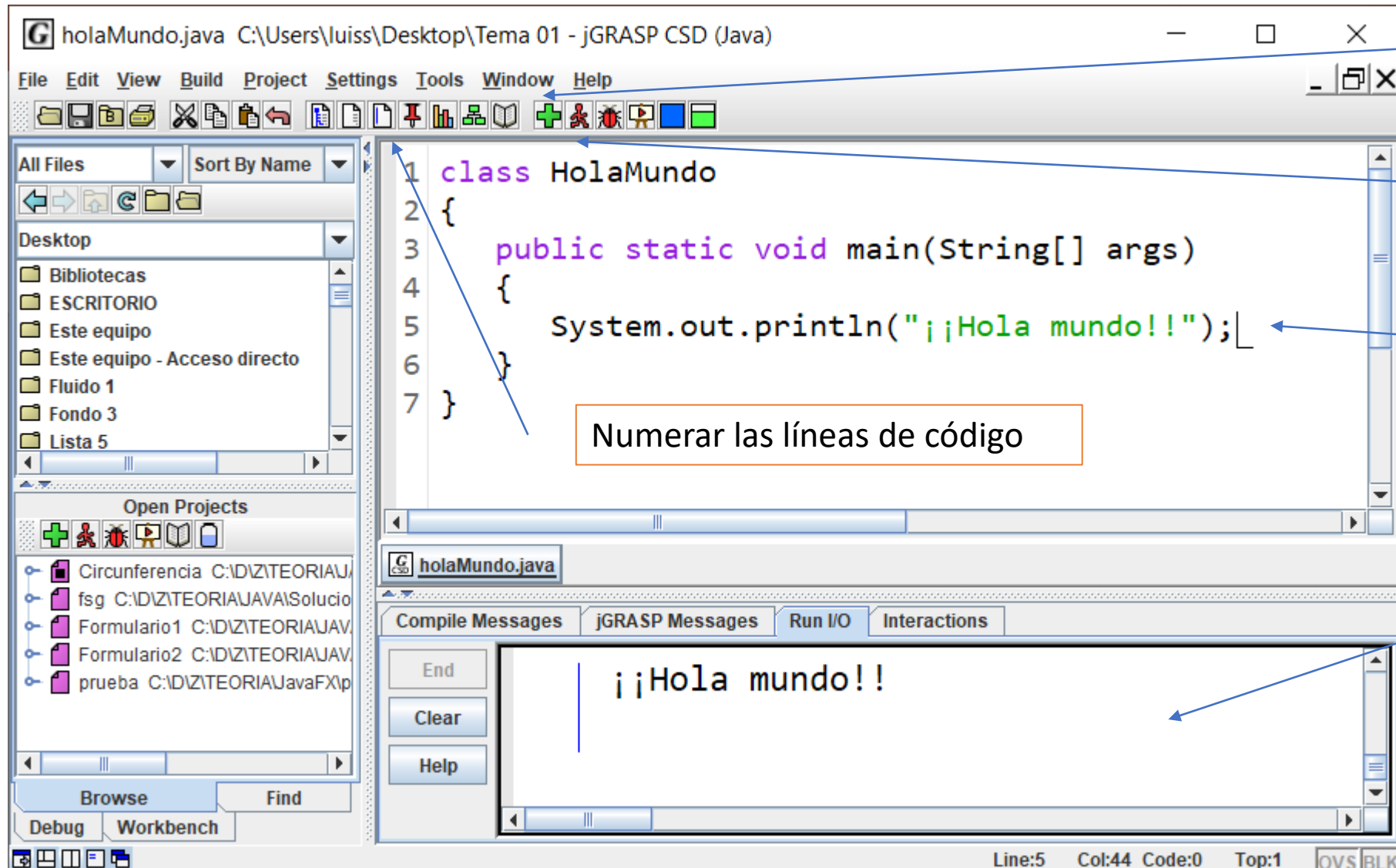
Se observa que el nombre del archivo y de la
clase es el mismo (como veremos deberá ser
un identificador)

Practicas Tema 01

HolaMundo.class

HolaMundo.java

Editar, compilar y ejecutar un programa



Botón para compilar

Botón para ejecutar

El código se escribe dentro del "main" de la clase "HolaMundo"

Numerar las líneas de código

Ventana de errores de compilación, mensajes, consola de ejecución y de interacción

Elementos del lenguaje

Objetivos

- 1) Saber definir identificadores
- 2) Conocer las palabras reservadas
- 3) Escribir comentarios en un programa

Identificadores

Un identificador es una palabra que representa elementos de un lenguaje de programación (variables, constantes, nombres de métodos, palabras reservadas, etc)

En java un identificador se define del siguiente modo:

- Comienza con una letra, un subrayado (_) o un símbolo de dólar (\$). Los siguientes caracteres pueden ser letras, dígitos y subrayado.
- Se distinguen las mayúsculas de las minúsculas.
- No hay una longitud máxima establecida para el identificador

Ejemplos de identificadores correctos:

- a) con1
- b) numero_de_hijos
- c) n2
- d) _contador
- e) Sumar

Ejemplos de identificadores incorrectos:

- a) 1alta
- b) numero de hermanos

Palabras reservadas

Las palabras reservadas son identificadores predefinidos que tienen un significado para el compilador y por tanto no pueden usarse como identificadores creados por el usuario en los programas.

Las palabras reservadas en Java ordenadas alfabéticamente son las siguientes:

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

Comentarios

En un código fuente o programa se pueden escribir comentarios en Java de forma que el compilador descartará tales líneas a la hora de la compilación. Hay de dos tipos:

- a) Comentarios en línea. Se ponen en una sola línea o a continuación de una línea de código.

```
// Comentario en línea
```

Se observa que el comentario en línea debe comenzar por “//”.

- b) Comentario de múltiples líneas.

```
/* Esto es un comentario  
en múltiples líneas */
```

Se observa que los comentarios en múltiples líneas comienzan por “/*” y terminan por “*/” .

Comentarios (II)

```
1  /*
2  AUTOR: Luis Fernández Ruiz
3  FECHA: 3/6/2013
4  */
5  class HolaMundo //Clase HolaMundo
6  {
7      public static void main(String[] args)
8      {
9          //Mostrar por consola ¡¡Hola mundo!!
10         System.out.println("¡¡Hola mundo!!");
11     }
12 }
```


Tipos de datos

Objetivos

- 1) Tipos primitivos numéricos: identificación y valores
- 2) Tipo primitivo booleano: identificación y valores
- 3) Tipo primitivo de carácter: identificación y valores
- 4) Tipo de cadena: identificación y valores

Tipos de datos primitivos numéricos

TIPO	BYTES	MÍNIMO	MÁXIMO
byte	1	-128	127
short	2	-32768	32767
int	4	-2147483648	2147483647
long	8	-9223372036854775808	9223372036854775807
float	8	1.17549435E-38	3.4028235E38
		-3.4028235E38	-1.17549435E-38
double	16	2.2250738585072014E-308	1.7976931348623157E308
		-1.7976931348623157E308	-2.2250738585072014E-308

TIPO: Entero byte, entero corto, entero, entero largo, real de simple precisión, real

Tipos de datos primitivos numéricos (II)

Ejemplos de literales o valores de tipos de datos primitivos numéricos:

- 1) Notación decimal: 56, -128, 0, 122.843, -0.034
- 2) Notación científica: 34e3, 45e-9, 1E10
- 3) Notación binaria: 0b1101, -0B101
- 4) Notación octal: 0170, -027
- 5) Notación hexadecimal: 0xA12, -0X1A
- 6) Notación con letra: 23L o 23l (es de tipo long), -0.001F o -0.001f (es de tipo float), 2.007D o 2.007d (es de tipo double)

Tipo de dato primitivo booleano

TIPO	BYTES	MÍNIMO	MÁXIMO
boolean	1	false	true

Ejemplos de literales o valores del tipo de dato primitivo booleano:

true, false

Tipo de dato primitivo carácter

TIPO	BYTES	MINIMO	MAXIMO
char	2	0	65536

Ejemplos de literales de tipo primitivo de carácter son los siguientes:

- 1) Notación numérica: 23, 0b101, 0B10, 070, 0x1A, 0XA1
- 2) Notación de comillas: 'a', '?', 'ñ', etc.
- 3) Notación de escape: '\n' (carácter de salto de línea), '\t' (carácter de tabulación), '\b' (carácter de emitir el sonido de una campana), '\\' (carácter \), '\0' (carácter NULL) etc.
- 4) Notación Unicode: '\u03A1'

Tipo de dato de cadena

En java se tiene el tipo de datos “String” para representar secuencia de caracteres.

Ejemplo de literales de cadenas son: "Hola mundo", "¿Cómo te llamas? ", etc

En los literales de cadenas se pueden poner caracteres de escape o unicode: "La ruta del archivo es:\tC:\\dir\\ficheros\\alumnos.txt\n"

Se observa que todas las cadenas se escriben entre comillas dobles.

Constantes y variables

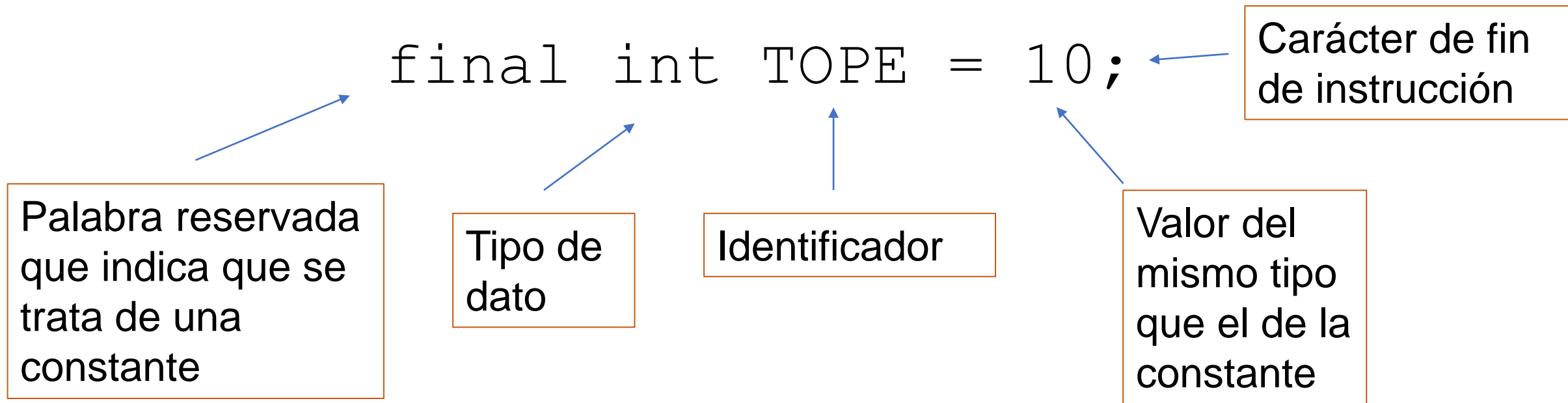
Objetivos

- 1) Saber qué es una constante
- 2) Saber qué es una variable
- 3) Saber declarar e inicializar constantes y variables de cada uno de los tipos de datos.

Constantes

Una constante es un valor de un tipo con un nombre que no se puede modificar durante la ejecución del programa

Instrucción de declaración e inicialización de una constante



Constantes (II)

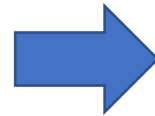
Instrucción de declaración e inicialización de múltiples constantes del mismo tipo

Identificador y valor de una nueva constante

`final double MINIMO = 3.5, MEDIO = 0.5, G = 9.8;`

Carácter separador de nueva constante

También se podía haber hecho lo siguiente



```
final double MINIMO = 3.5;  
final double MEDIO = 0.5;  
final double G = 9.8;
```

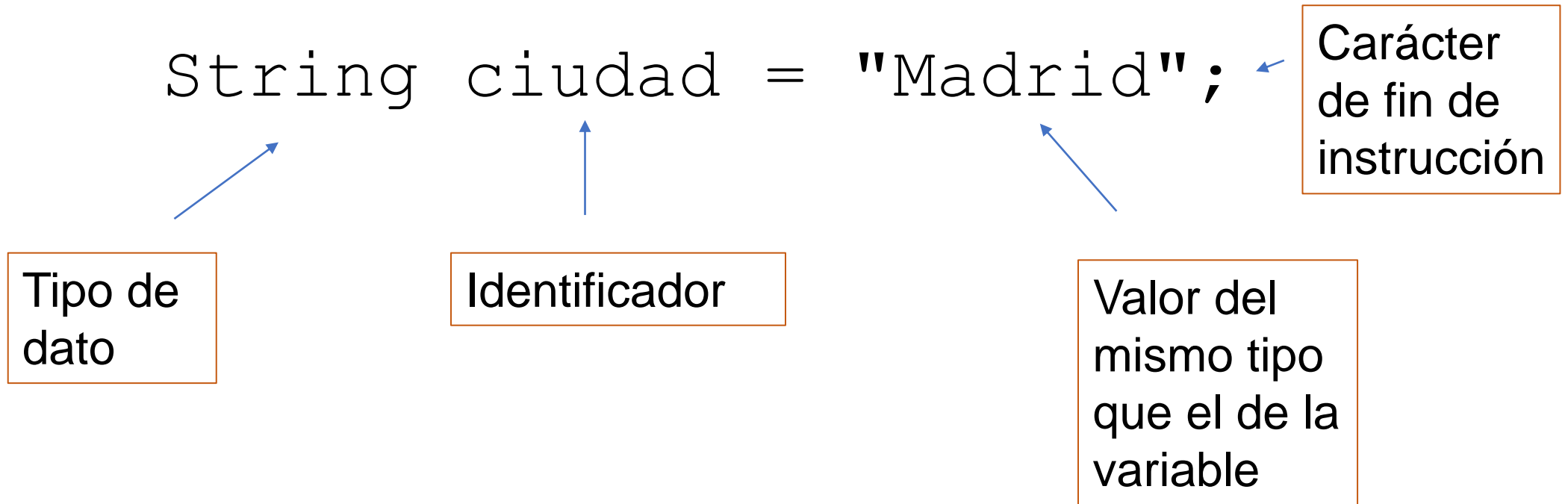
Ejemplos de constantes

```
1 class Constantes
2 {
3     public static void main(String[] args)
4     {
5         //Constantes
6         final int MAXIMO = 5;
7         final float PESO_MINIMO = 3.5f;
8         final String PAIS = "España";
9         final char LETRA_A = 'A';
10        final byte UNO = 1, DOS = 2;
11        final double N1 = 0.1, N2 = 0.2, N3 = 0.3;
12    }
13 }
```

Variables

Una variable es un valor de un tipo con un nombre cuyo valor se puede modificar durante la ejecución del programa

Instrucción de declaración e inicialización de una variable



Variables (II)

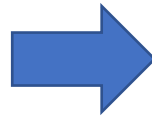
Instrucción de declaración e inicialización de múltiples variables del mismo tipo

Identificador y valor de una nueva variable

```
float a = 1.2f, b = 0, c = 0.5F;
```

Carácter separador de nueva variable

También se podía haber hecho lo siguiente



```
float a = 1.2f;  
float b = 0;  
float c = 0.5F;
```



Ejemplo de Constantes y variables

- Habitualmente primero se definen las constantes y luego las variables al inicio del método "main"
- No puede haber dos identificadores de constantes o variables con el mismo nombre en el mismo bloque de código

```
class Variables
{
    public static void main(String[] args)
    {
        //Constantes
        final long N = 10000000000L;

        //Variables
        int contador = 0;
        int suma = 0;
        byte a = 0, b = 1, c = 2;
    }
}
```

A partir de ahora sólo se escribirá el contenido del método main

Salida y entrada de datos

Objetivos

- 1) Mostrar datos por consola saltando de línea
- 2) Mostrar datos por consola sin saltar de línea
- 3) Mostrar datos con formato
- 4) Lectura de datos
- 5) Detener control, dormir y limpiar la consola

Salida por consola

Se puede mostrar por consola valores, constantes y variables.

Instrucción de salida por consola de un dato saltando de línea

```
//a vale 100 y UNO, 1.0
```

```
System.out.println(1000);
```

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

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

Salida
consola

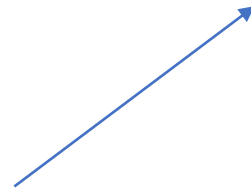


1000

100

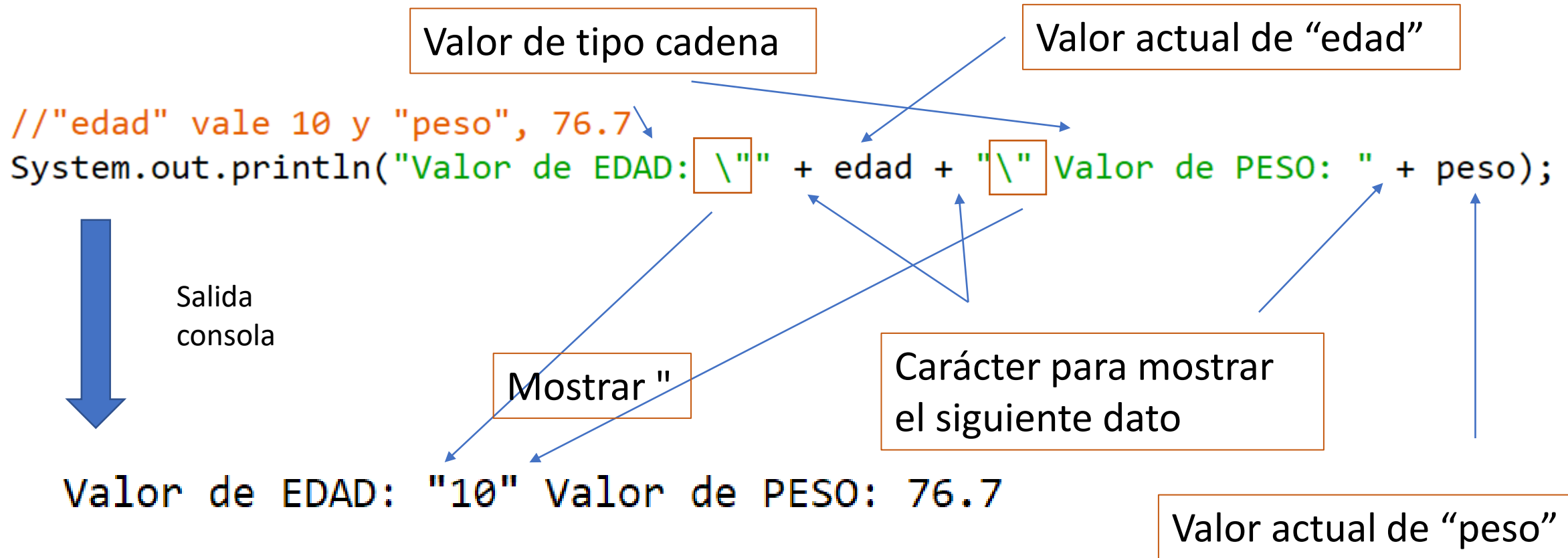
1.0

Se escribe un valor, una
constante o una variable



Salida por consola (II)

Instrucción de salida por consola de varios datos saltando de línea





Ejemplo de Salida por consola

Hágase un programa en el que se defina una constante con tu nombre y una variable con tu edad. Se mostrará por consola dichos datos en líneas distintas con el texto “Me llamo “X”” (el nombre se muestra entrecomillado) y “Tengo X años”

```
//Constantes
```

```
final String NOMBRE = "Pedro";
```

```
//Variables
```

```
int edad = 23;
```

```
//Programa
```

```
System.out.println("Me llamo \""+NOMBRE+"\"");
```

```
System.out.println("Tengo " + edad + " años");
```

Salida
consola



Me llamo "Pedro"
Tengo 23 años

Salida por consola (III)

Instrucción de salida por consola de uno más datos sin saltar de línea

```
// "nombre" vale "Pedro"; veces, 10 y tiempo, 23.7  
System.out.print("Nombre: " + nombre + "\n" + "veces = " + veces +  
" tiempo = " + tiempo);
```



Salida
consola

Carácter de salto de línea

Nombre: Pedro
veces = 10 tiempo = 23.7

Siguiente carácter a escribir

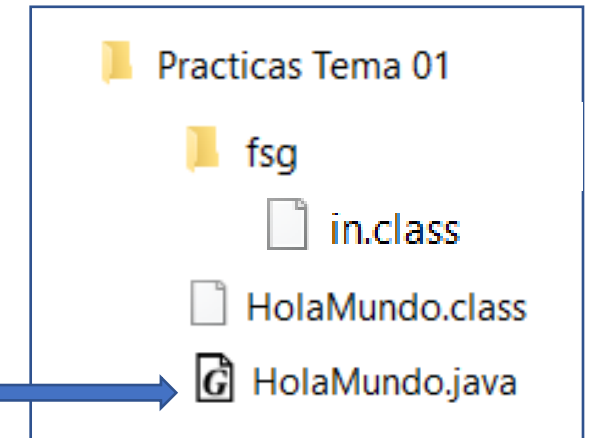
Entrada de datos

- Se utilizará una **librería de usuario** para leer datos de entrada por consola.
- Las **librerías de usuario** son archivos “.class” que se almacenan en un directorio con métodos para hacer determinadas acciones.
- En el caso de la lectura de datos por consola se utilizará **la librería “in”** que permite leer y validar datos que se introducen por consola, detener el control, suspender hilos y ejecutar procesos (como limpiar la pantalla).
- Para poder acceder a dicha librería se crea en el directorio de la aplicación una carpeta “fsg” con el archivo “in.class”.

En cada aplicación “.java”
con entrada de datos se
pondrá como primera línea

```
import fsg.in;
```

Importar libreria



Entrada de datos por consola (II)

Instrucción para leer un dato entero con un mensaje estándar

```
// "a" es de tipo int  
a = in.leerInt();
```



Salida
consola

Mensaje estándar para un entero

```
ESCRIBE UN ENTERO: a  
Valor incorrecto o desbordado  
ESCRIBE UN ENTERO: 67777777777777777777  
Valor incorrecto o desbordado  
ESCRIBE UN ENTERO: 23
```

Se introduce un dato no numérico o con un valor fuera del rango válido

Se muestra un mensaje estándar de error

Se introduce un dato válido y se almacena en "a"

Entrada de datos por consola (III)

Instrucción para leer un dato entero con un mensaje específico

```
// "a" es de tipo int
a = in.leerInt("EDAD: ");
```

Se tienen métodos leerByte, leerShort, leerLong, leerFloat, leerDouble, leerBoolean, leerChar, leerString y leerLine para introducir datos de tipo byte, short, long, float, double, boolean, char, String (palabra) y String (línea)



Salida
consola

Mensaje específico

[illegible]

Se introduce un dato no numérico o numérico fuera del rango válido.

Se muestra un mensaje estándar de error

Se introduce un dato válido y se almacena en “a”



Ejemplo de Entrada por consola

Hágase un programa en el que se introduzca el nombre y apellidos de una persona y su edad. El programa mostrará por consola dichos datos.

```
//Variables  
String nombre = "", apellidos = "";  
double peso = 0;
```

```
//Programa  
nombre = in.leerString("Escribe tu nombre: ");  
apellidos = in.leerLine("Escribe tus apellidos: ");  
peso = in.leerDouble("Escribe tu peso: ");
```

```
System.out.println("Me llamo " + nombre + " " + apellidos);  
System.out.println("Peso " + peso + " Kg");
```

```
Escribe tu nombre: Luis  
Escribe tus apellidos: González Sanz  
Escribe tu peso: 67,3  
Me llamo Luis González Sanz  
Peso 67.3 Kg
```

Salida
consola

Los datos reales se
introducen con ","

Entrada de datos por consola (IV)

Instrucción para leer una constante

//Constantes

```
final double PESO_MAXIMO;
```

Se declara la constante sin inicializar

//Programa

```
PESO_MAXIMO = in.leerDouble("PESO MAXIMO: ");
```

Se lee el valor de la constante

Salida
consola

PESO MAXIMO: 56,7

Se lee el valor y se almacena en la constante
"PESO_MAXIMO"

Métodos de la librería “in”

Además de los métodos de entrada definidos en la librería “in”, se tienen también los siguientes métodos:

```
//Detiene el control hasta que  
// se pulsa enter  
in.detener();
```



Pulsa enter para continuar ...

Salida
consola

```
//Detener el control por un tiempo  
in.dormir(2000);
```

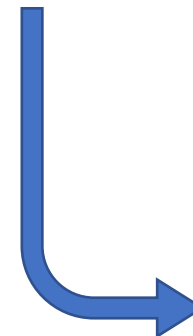


Durante 2000
milisegundos se detiene la
ejecución del programa

```
//Limpiar la consola  
in.cls();
```



Para ver el efecto de dicho
método se tiene que activar la
consola de MSDOS en JGrasp



Salida de datos formateada

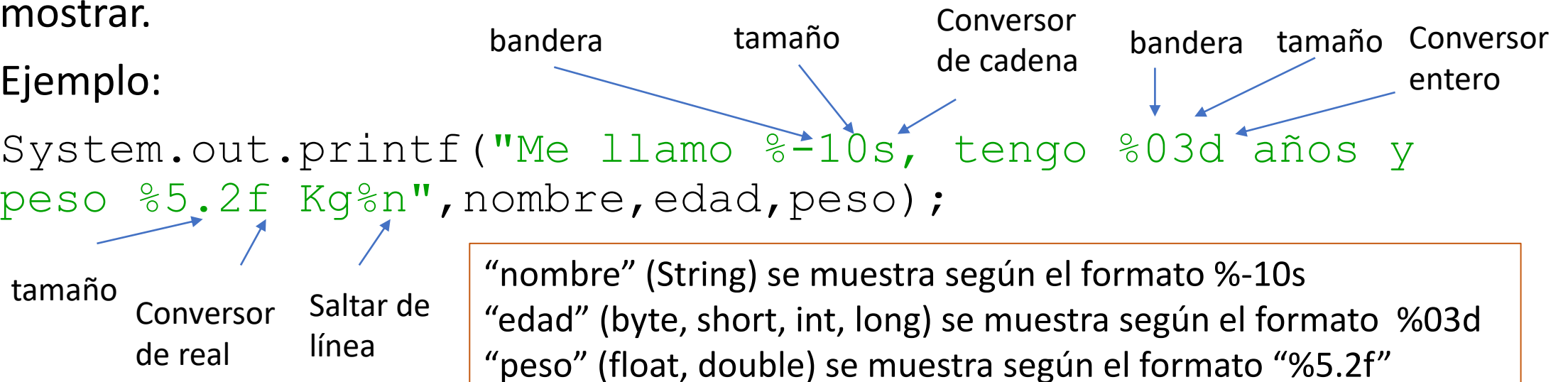
Instrucción de salida de datos formateada

```
System.out.printf("Cadena de formato", v1, ... , vn);
```

La cadena de formato son textos en los que se intercala **un elemento de formato** del tipo **%[Banderas][Tamaño]Conversor** para cada uno de los valores **v1, ... , vn** a mostrar.

Ejemplo:

```
System.out.printf("Me llamo %-10s, tengo %03d años y  
peso %5.2f Kg%n", nombre, edad, peso);
```



Salida de datos formateada: Conversores

Banderas	Tamaño	Conversor	Tipo valor
0	n	o	byte
+		h	short
(d	int
-		x	long
-	n	b	boolean
-	n	c	char
0	n	a	float double
+		e	
(1.m	f	
-	n.m	g	
,	n	s	String
-		n	
		%	

%[Banderas][Tamaño]Conversor

Conversores habituales

Conversor (entero, booleano, carácter y cadena)

o El valor entero se muestra en octal.

h/H El valor entero se muestra en hexadecimal (no permite la bandera 0)

x/X El valor entero se muestra en hexadecimal

d El valor entero se muestra en base 10.

b/B Conversor para el tipo primitivo boolean.

c/C Conversor para el tipo primitivo char

s/S Conversor para cadenas

Conversor en mayúsculas → Las letras que contenga el valor se muestran en mayúsculas

Salida de datos formateada: Conversores (II)

Banderas	Tamaño	Conversor	Tipo valor
0	n	o	byte
+		h	short
(d	int
-		x	long
-	n	b	boolean
-	n	c	char
0	n	a	float double
+		e	
(1.m	f	
-	n.m	g	
,	n	s	String
-		n	
		%	

%[Banderas][Tamaño]Conversor

Conversores habituales

Conversor (real y especiales)

a/A El valor real se muestra en hexadecimal

e/E El valor real se muestra en notación científica

f El valor real se muestra con notación decimal

g/G El valor real se muestra en notación científica o decimal

n Salto de línea (no admite banderas ni tamaño, y no tiene asociado ningún valor)

% Muestra el '%'

Conversor en mayúsculas → Las letras que contenga el valor se muestran en mayúsculas

Salida de datos formateada: Tamaños

Banderas	Tamaño	Conversor	Tipo valor
0 + (-	n	o h d x	byte short int long
-	n	b	boolean
-	n	c	char
0 + (- ,	n 1.m n.m	a e f g	float double
-	n	s	String
		n	
		%	

Tamaño

n El valor ocupará un total de n espacios (si el valor ocupase más espacios, se mostraría en su totalidad).

n.m El valor ocupará n espacios y se mostrará con m decimales. Puede ser que n sea menor que m. El valor real se muestra redondeado a “m” decimales.

%[Banderas][**Tamaño**]Conversor

← Opcional

Salida de datos formateada: Banderas

Banderas	Tamaño	Convertor	Tipo valor
0 + (-	n	o h (no bandera 0) d x	byte short int long
-	n	b	boolean
-	n	c	char
0 + (- ,	n 1.m n.m	a e f g	float double
-	n	s	String
		n	
		%	

%[Banderas][Tamaño]Convertor

← Opcional

Banderas

0 Rellenar con 0 por la izquierda hasta completar el tamaño.

+ Se mostrará el signo + para los positivos. Sólo válido para tipos primitivos numéricos.

(Los valores negativos se pondrán entre paréntesis y no se mostrará el signo '-'.

, Se mostrará el carácter decimal correspondiente a la zona geográfica.

- Se ajustará el valor a la izquierda (por defecto se ajusta a la derecha)

Ejemplo de Salida de datos formateada



```
// "nombre" vale "Pedro"; "edad", 13 y "peso", 20.1  
System.out.printf("Me llamo *%-10s*, tengo %03d  
años y peso #%.2f# K%n", nombre, edad, peso);
```



Salida
consola

Me llamo *Pedro *, tengo 013 años y peso # 20,10# Kg

"Pedro" se muestra ocupando 10 espacios y alineado a la izquierda

13 se muestra ocupando 3 espacios y rellenando con 0 por la izquierda

20.1 se muestra ocupando 7 espacios con 2 decimales y alineado a la derecha

Asignación de variables y algoritmos

Objetivos

- 1) Asignación de variables del mismo tipo
- 2) Algoritmos
- 3) Conversión de tipos primitivos numérico

Asignación de variables

Instrucción de asignación entre variables del mismo tipo


La instrucción de asignación permite cambiar el valor de una variable asignándole el valor de otra variable del mismo tipo.

```
// "a" y "b" son enteras  
a = b;
```

A blue arrow points from the variable 'b' in the code 'a = b;' to the variable 'a'. Another blue arrow points from the variable 'a' to the assignment operator '=', indicating the direction of data flow.

A la variable "a" se le asigna el valor que tenga "b"
Copia "b" en "a" (se pierde el valor de "a")

```
// "minimo" y "peso" son reales  
minimo = peso;
```

A blue arrow points from the variable 'peso' in the code 'minimo = peso;' to the variable 'minimo'. Another blue arrow points from the variable 'minimo' to the assignment operator '=', indicating the direction of data flow.

A la variable "minimo" se le asigna el valor que tenga "peso"
Copia "peso" en "minimo" (se pierde el valor de "mínimo")

```
// "nombre" y "dato" son de tipo cadena  
nombre = dato;
```

A blue arrow points from the variable 'dato' in the code 'nombre = dato;' to the variable 'nombre'. Another blue arrow points from the variable 'nombre' to the assignment operator '=', indicating the direction of data flow.

A la variable "nombre" se le asigna el valor que tenga "dato"
Copia "dato" en "nombre" (se pierde el valor de "nombre")

Algoritmos

Un **algoritmo** es un conjunto finito de instrucciones o pasos que **sirven para** ejecutar una tarea o resolver un problema.

Ejemplos de pasos en un algoritmo:

```
double a = 1.2, b = 1.2;
```

1. Declarar e inicializar dos variables reales “a” y “b” con el valor 1.2
2. Leer un dato real (“Escribe un real: ”) y almacenarlo en la variable “a”
3. Mostrar saltando de línea el valor de “a”, un espacio, el valor de “b”
4. Copiar “dato” en “peso”

```
peso = dato;
```

```
System.out.println(a + " " + b);
```

```
a = in.leerDouble("Escribe un real: ");
```

Ejemplo de algoritmo

Hágase un algoritmo que lea dos variables enteras, intercambie sus valores y las muestre en el orden como se introdujeron.

1. Declarar e inicializar tres variables enteras “a”, “b” y “aux” con el valor 0
2. Leer la variable “a” con un mensaje del tipo “Escribe un entero: ”
3. Leer la variable “b” con un mensaje del tipo “Escribe un entero: ”
4. Copiar “a” en “aux” (se tiene una copia de “a”)
5. Copiar “b” en “a”
6. Copiar “aux” (antiguo valor de “a”) en “b”
7. Mostrar saltando de línea el valor de “a”, un espacio, el valor de “b”

n. Intercambiar “a” y “b”
(puede considerarse como un paso de un algoritmo)

Traducir a
instrucciones





Ejemplo de Algoritmo (II)

```
//Variables
```

```
int a = 0, b = 0, aux = 0;
```

```
//Programa
```

```
a = in.leerInt("Escribe un entero: ");
```

```
b = in.leerInt("Escribe un entero: ");
```

```
aux = a;
```

```
a = b;
```

```
b = aux;
```

```
System.out.println(a+" "+b);
```



```
Escribe un entero: 45  
Escribe un entero: 23  
23 45
```

Conversión de tipos primitivos numéricos

Instrucción de asignación de variables de distintos tipos numéricos

Escala de tipos numéricos de mayor a menor rango de valores

```
// "peso" es double  
// "edad" es int
```

double	float	long	int	short	byte
				char	

Rango de "peso" > Rango de "edad"

```
peso = edad;
```

Se copia en "peso" el valor actual de "edad" que se convierte a double (no hay pérdida de datos)

Operadores de conversión explícita: (double), (float), (long), (int), (short), (byte), (char)

Rango de "edad" < Rango de "peso"

```
edad = peso; ➡ edad = (int) peso;
```



```
error: incompatible  
types: possible  
lossy conversion  
from double to int
```

Se copia en "edad" el valor actual de "peso" que se convierte a entero con el **operador de conversión explícita (int)** quitando los decimales

Ejemplo de Conversor de tipo numérico



Hágase un programa en el que se lea un entero largo, se copie a un entero y a un carácter, y se muestren los valores copiados.

```
//Variables
```

```
long a = 0;
```

```
int valor = 0;
```

```
char letra = '\0';
```

```
//Programa
```

```
a = in.leerLong();
```

```
valor = (int) a;
```

```
letra = (char) a;
```

```
System.out.println(valor+" "+letra);
```

Salida
consola



```
ESCRIBE UN ENTERO LARGO: 89  
89 Y
```

```
ESCRIBE UN ENTERO LARGO: 9999999999999999  
276447231 ?
```

```
ESCRIBE UN ENTERO LARGO: -1  
-1 ?
```

```
ESCRIBE UN ENTERO LARGO: -9999999999999999  
-276447231 ?
```

Expresiones y operadores

Objetivos

- 1) Concatenar cadenas
- 2) Expresiones aritméticas. Truncar y redondear
- 3) Comparaciones numéricas y de cadenas
- 4) Expresiones lógicas. Validación de datos de entrada.
- 5) Operadores de bits
- 6) Operadores de acumulación y condicional
- 7) Tabla de orden de prioridad entre operadores. Operador !
- 8) La librería Math: Números aleatorios



Concatenar cadenas

OPERADOR	USO	OPERACIÓN
+	a+b	Concatenar

Renault Clio 0000 A

Salida
consola

```
String marca = "Renault";  
String modelo = "Clio";  
String matricula = "0000 A";  
String dato = "";  
dato = marca + " " + modelo + " " + matricula;  
System.out.println(dato);
```

```
int a = 1, b = 2;  
String cad = "";
```

```
cad = a + " " + b;
```

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

Salida
consola

1 2

Expresiones

Una **expresión** es un cálculo que se realiza obteniendo un valor de un tipo determinado.

- Una expresión supone evaluar uno o más operadores en un orden determinado y preciso.
- Las expresiones no se pueden escribir directamente en el programa. Aparecen formando parte de una instrucción.

Ejemplos de expresiones:

<code>a+b*2</code>	<code>n%2==0 && n<=10</code>	<code>a+=veces</code>
<code>numero>4</code>	<code>valor>>3</code>	<code>-contador</code>
<code>valor++</code>	<code>a/(double)b</code>	<code>pro = a*b*c</code>

Expresiones aritméticas

Se tienen los siguientes operadores aritméticos que operan sobre un solo dato: ++, --, -

OPERADOR	USO	OPERACION
++	a++ ++a	Devuelve el valor de “a” y luego incrementa “a” en 1 Incrementa “a” en 1 y devuelve el nuevo valor de “a”
--	a--	Devuelve el valor de “a” y luego decrementa “a” en 1 Decrementa “a” en 1 y devuelve el nuevo valor de “a”
-	-a	Devuelve el valor de “a” con el signo cambiado

Ejemplos de uso de ++, -- y -

Instrucción para incrementar una variable numérica en 1

[illegible]

Instrucción para decrementar una variable numérica en 1

$$\dot{I} \dashrightarrow; \quad 0 \quad \dashrightarrow \dot{I};$$

Instrucción para cambiar el signo de una variable numérica

```
valor = -valor;
```

Ejemplos de ++, -- y -

Hágase un programa que lea un entero, lo incremente en una unidad, lo muestre por consola, lo cambie de signo y lo vuelva a mostrar

```
//Variables
```

```
int n = 0;
```

```
//Programa
```

```
n = in.leerInt();
```

```
n++;
```

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

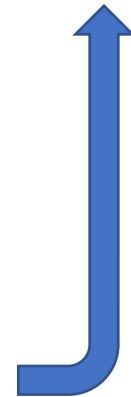
```
n = -n;
```

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

ESCRIBE UN ENTERO: -87

-86

86



Salida
consola

Asignación con operadores ++ y --

Instrucción que incrementa en 1 el valor una variable numérica y el nuevo valor de esa variable se asigna a otra variable del mismo tipo

`valor = ++dato;`  `dato++;
valor = dato;`

Instrucción que asigna a una variable numérica el valor de otra variable del mismo tipo y luego incrementa el valor de esta última variable en 1

`valor = dato++;`  `valor = dato;
dato++;`

Expresiones aritméticas binarias

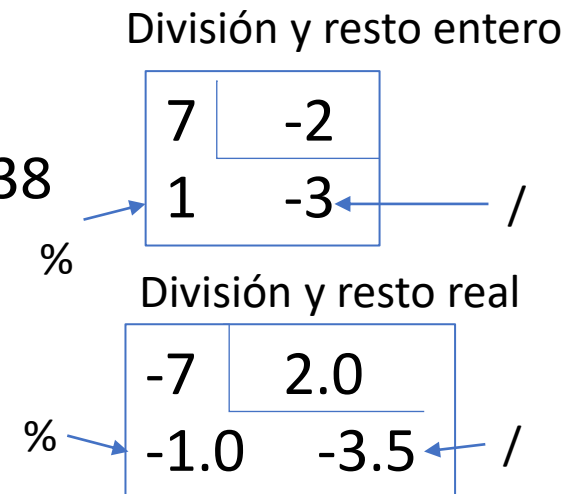
OPERADOR	USO	OPERACION
+	a+b	Devuelve la suma de los valores de “a” y “b”.
-	a-b	Devuelve la resta de los valores de “a” y “b”.
*	a*b	Devuelve el producto de “a” y “b”
/	a/b	Devuelve el cociente entero o real entre “a” y “b”
%	a%b	Devuelve el resto entero o real entre “a” y “b”

Ejemplos con valores double, float (F), long (L) e int:

$3 + 2 \rightarrow 5$ $3.1 + 2 \rightarrow 5.1$ $3 - 8L \rightarrow -5L$ $3.1F * 1.98 \rightarrow 6.138$

$7 / 2L \rightarrow 3L$ $7 \% 2 \rightarrow 1$ $7 / 2.0 \rightarrow 3.5$ $7F \% 3.2F \rightarrow 0.6F$

$7 / (\text{double}) 2 \rightarrow 3.5$ $7 \% (\text{double}) 2 \rightarrow 1.0$



Ejemplos de uso de +, -, *, / y %

Instrucción que obtiene un cálculo

```
// "coste", "precio" double, "unidades" int
coste = precio * unidades;
```

Instrucción de conversión de datos del mismo tipo

```
// "edad", "dias" int
dias = edad * 365;
```

Instrucción que modifica una variable

```
// "a" float // "edad" int
a = 2 * a;    edad = 3 + edad;
```

Instrucción que acumula una variable

```
// "coste" y "total" double
total = total + coste;
```

```
// "n", "u" int
u = n % 10;
```

Obtiene las unidades de "n".
Si n = 1084, entonces u = 4

```
// "seg", "min" int
min = seg / 60;
```

Obtiene los minutos que hay en "seg". Si seg = 125, entonces min = 2

```
// "seg" int
seg = seg % 60;
```

Descuenta de "seg" los minutos que contenga.
Si seg = 125, entonces seg = 5

```
// "n" int
n = n / 10;
```

Quita el dígito de la derecha. Si n = 1094, entonces n = 109



Expresiones aritméticas binarias

Hágase un programa que lea una letra en minúsculas y la muestre en mayúsculas. No se pueden introducir vocales con acento ni diéresis, ni tampoco la letra 'ñ'.

```
//Variables
```

```
char letraMinuscula = '\0', letraMayuscula = '\0';  
int codigoAscii = 0, offset = 0;
```

```
// Programa
```

```

letraMinuscula = in.leerChar("Escribe una letra en minúscula: ");
offset = letraMinuscula - 'a';           Escribe una letra en minúscula: f
codigoAscii = 'A'+offset;               La letra en mayúsculas es F Salida
letraMayuscula = (char) codigoAscii;      ↑ Consola
System.out.println("La letra en mayúsculas es " + letraMayuscula);

```

Evaluar múltiples operadores aritméticos

En un cálculo pueden **aparecer dos o más operadores aritméticos**. La forma de evaluar tal expresión sería la siguiente:

ORDEN	OPERADORES		ASOCIATIVIDAD	(), 2, 3	Izquierda a derecha	
1	++a, --a, (), -a, (tipo)		Ejemplos.	$a + b - c$	$a++ + --c$	$(\text{int}) d * 4$
2	* / %		Sean las variables: int a = 5, b = 6, c = 3; double d = 2.3,	$5 + 6 - 3$ $11 - 3$ 8	$5++ + --3$ $5+2$ 7 (a=6 c=2)	$(\text{int}) 2.3 * 4$ $2 * 4$ 8
3	+ -					
<div><div><div>$a * (b - 4) + c$</div><div>$5 * (6 - 4) + 3$</div><div>$5 * 2 + 3$</div><div>$10 + 3$</div><div>13</div></div><div><div>$(a + 11) \% b / c$</div><div>$(5 + 11) \% 6 / 3$</div><div>$16 \% 6 / 3$</div><div>$4 / 3$</div><div>1</div></div><div><div>$(a + 11) \% (b / c)$</div><div>$(5 + 11) \% (6 / 3)$</div><div>$16 \% (6 / 3)$</div><div>$16 / 2$</div><div>0</div></div></div>						
<div><div><div>$a + d * b$</div><div>$5 + 2.3 * 6$</div><div>$5 + 2.3 * 6.0$</div><div>$5 + 13.8$</div><div>$5.0 + 13.18$</div><div>18.18</div></div><div><div>$(\text{int}) (d * 4)$</div><div>$(\text{int}) 2.3 * 4$</div><div>$(\text{int}) 2.3 * 4.0$</div><div>$(\text{int}) 9.2$</div><div>9</div></div><div><div>$-(1 - a) * 2$</div><div>$-(1 - 5) * 2$</div><div>$- -4 * 2$</div><div>$4 * 2$</div><div>8</div></div><div><div>$c * (--d + 4.1)$</div><div>$3 * (--2.3 + 4 - 1)$</div><div>$3 * (1.3 + 4.1)$</div><div>$3 * 5.4$</div><div>$3.0 * 5.4$</div><div>16.2 (d=1.3)</div></div></div>						



Truncar y redondear

Expresiones para truncar y redondear una variable real o entera

A LAS	TRUNCAR d	3594,2785	REDONDEAR d	3594,2785	TIPO
unidades de millar	$(\text{int}) (d/1000) * 1000$	3000	$(\text{int}) (d/1000+0.5) * 1000$	4000	int
centenas	$(\text{int}) (d/100) * 100$	3500	$(\text{int}) (d/100+0.5) * 100$	3600	int
decenas	$(\text{int}) (d/10) * 10$	3590	$(\text{int}) (d/10+0.5) * 10$	3590	int
unidades	$(\text{int}) d$	3594	$(\text{int}) (d+0.5)$	3594	int
décimas	$(\text{int}) (d*10) / 10.0$	3594,2	$(\text{int}) (d*10 + 0.5) / 10.0$	3594,3	double
centésimas	$(\text{int}) (d*100) / 100.0$	3594,27	$(\text{int}) (d*100 + 0.5) / 100.0$	3594,28	double
milésimas	$(\text{int}) (d*1000) / 1000.0$	3594,278	$(\text{int}) (d*1000+0.5) / 1000.0$	3594,278	double

Ejemplo de Truncar y redondear

Hágase un programa que lea un número real y lo trunque a las centenas y lo redondee a las décimas

```
//Variables
```

```
double dato = 0;
```

```
double decimas = 0;
```

```
int centenas = 0;
```

```
//Programa
```

```
dato = in.leerDouble();
```

```
centenas = (int) (dato/100) * 100;
```

```
decimas = (int) (dato*10 + 0.5) / 10.0;
```

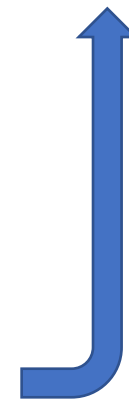
```
System.out.println("TRUNCADO A LAS CENTENAS: "+centenas);
```

```
System.out.println("REDONDEADO A LAS DECIMAS: "+decimas);
```

ESCRIBE UN REAL: 1845,35009

TRUNCADO A LAS CENTENAS: 1800

REDONDEADO A LAS DECIMAS: 1845.4



Salida
consola



Ejemplos de múltiples operadores aritméticos

Ejemplo de **instrucciones** en las que aparecen múltiples operadores aritméticos serían:

```
// "compra", "precio" y "iva" double;  
// "unidades" int  
compra = unidades * precio + iva;
```

```
// "media" double  
// "nota1" y "nota2" int  
media = (nota1 + nota2) / 2.0;
```

```
// "a", "b" int  
// "r" double  
r = 1 + a / (double)b;
```

```
// "a" int "b" double  
System.out.printf("%d %f\n", (int) (2*a + ++b), b);
```

```
// "a", "b" int  
System.out.println("SUMA = " + (a + b) );  
System.out.println("PRODUCTO = " + a * b);
```

a + b debe ponerse entre paréntesis para hacer la operación antes de concatenar los datos

Ejemplos de múltiples operadores aritméticos (II)

Hágase un programa que lea las dimensiones en metros de un deposito y obtenga su capacidad en litros ($1 \text{ m}^3 = 1000 \text{ litros}$)

```
//Variables  
int ancho = 0, alto = 0, largo = 0;
```

```
//Programa  
ancho = in.leerInt("Ancho: ");  
alto = in.leerInt("Alto: ");  
largo = in.leerInt("Largo: ");
```

```
System.out.println("Capacidad: "+ ancho*alto*largo*1000 + " litros");
```

Ancho: 4
Alto: 3
Largo: 8
Capacidad: 96000 litros



Salida
consola

Ejemplos de Múltiples operadores aritméticos (III)

Hágase un programa que lea tres notas y obtenga la media redondeada a 1 decimal.

```
//Variables
```

```
int nota1 = 0, nota2 = 0, nota3 = 0;  
double media = 0;
```

```
//Programa
```

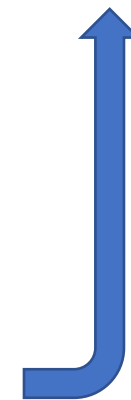
```
nota1 = in.leerInt("Escribe una nota: ");  
nota2 = in.leerInt("Escribe una nota: ");  
nota3 = in.leerInt("Escribe una nota: ");
```

```
media = (nota1+nota2+nota3)/3.0;
```

```
media = (int) (media*10)/10.0;
```

```
System.out.println("Media de las notas: "+media);
```

```
Escribe una nota: 7  
Escribe una nota: 5  
Escribe una nota: 8  
Media de las notas: 6.6
```



Salida
consola



Expresiones de comparación numéricas

OPERADOR	USO	OPERACION
<	a<b	Devuelve true si “a” es menor que “b”; si no, false
<=	a<=b	Devuelve true si “a” es menor o igual que “b”; si no, false
>	a>b	Devuelve true si “a” es mayor que “b”; si no, false
>=	a>=b	Devuelve true si “a” es mayo o igual que “b”; si no, false
==	a==b	Devuelve true si “a” es igual a “b”; si no, false
!=	a!=b	Devuelve true si “a” es distinto a “b”; si no, false

Ejemplos con valores double, float (F), long (L) e int:

3 > 2 → true 3.1 < 2.0 → false 4 == 4L → true 3.2F != 3.2 → false

Expresiones de comparación y aritméticas

En una expresión de comparación pueden aparecer a ambos lados del operador de comparación expresiones aritméticas. Se evalúa primero las expresiones aritméticas y por último se comparan los resultados

Ejemplos. Sean las variables: `int a = 5, b = 6; double d = 2.3;`

`2 * a + b > d + 1`

`2 * 5 + 6 > 2.3 + 1`

`10 + 6 > 2.3 + 1. 0`

`16 > 4.3`

`16. 0 > 4.3`

`true`

`(int) (d * a) == ++b + 9`

`(int) (2.3 * 5) == ++6 + 9`

`(int) (2.3 * 5.0) == 7 + 9`

`(int) 16.5 == 16`

`16 == 16`

`true`



Condiciones de comparación

- “a” es par $\rightarrow a \% 2 == 0$
- “a” es impar $\rightarrow a \% 2 == 1$
- “a” es múltiplo de 5 $\rightarrow a \% 5 == 0$
- “a” es divisible entre “b” $\rightarrow a \% b == 0$

Hágase un programa que lea un entero y muestre por consola verdadero si es par y falso, en caso contrario

```
//Variables
```

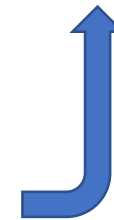
```
int n = 0;
```

```
//Programa
```

```
n = in.leerInt();
```

```
System.out.println(n % 2 == 0);
```

ESCRIBE UN ENTERO: 7
false



Salida
consola



Comparar cadenas

Para comparar dos cadenas se utilizan los métodos **equals** y **compareTo** de las cadenas.

FUNCIÓN	USO	OPERACIÓN
equals	a.equals(b)	"a" es igual "b"
compareTo	a.compareTo(b)>0	"a" es mayor que "b"
	a.compareTo(b)<0	"a" es menor que "b"
	a.compareTo(b)<=0	"a" es menor o igual que "b"
	a.compareTo(b)>=0	"a" es mayor o igual que "b"
!equals	! a.equals(b)	"a" es distinta a "b"

Ejemplos:

`"hola".equals("adios") -> false` `"hola".compareTo("adios") <= 0 -> false`

`!"hola".equals("adios") -> true` `"hola".compareTo("adios") > 0 -> true`

Comparar cadenas (II)

Hágase un programa que lea una palabra y compruebe que:

- no es igual a “hola”
- “hola” es anterior en orden alfabético.

```
//Variables
```

```
String cadena = "";
```

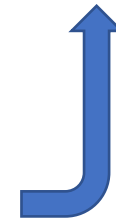
```
//Programa
```

```
cadena = in.leerString();
```

```
System.out.println(!cadena.equals("hola"));
```

```
System.out.println("hola".compareTo(cadena)<0);
```

```
ESCRIBE UNA PALABRA: Hola  
true  
false
```



Salida
consola



Operadores lógicos

OPERADOR	USO	OPERACIÓN
&&	a && b	Devuelve "true" si "a" y "b" son true; si no, false
&	a & b	
	a b	Devuelve "false" si "a" y "b" son false; si no, true
	a b	
!	!(a)	Devuelve true si "a" es false y false, si "a" es true
^	a^b	Devuelve false si "a" y "b" son false o true; si no, true.

Tabla Y (corto)

&&	T	F
T	T	F
F	F	

Tabla Y

&	T	F
T	T	F
F	F	F

Tabla O (corto)

	T	F
T	T	
F	T	F

Tabla O

	T	F
T	T	T
F	T	F

Tabla Ó

^	T	F
T	F	T
F	T	F

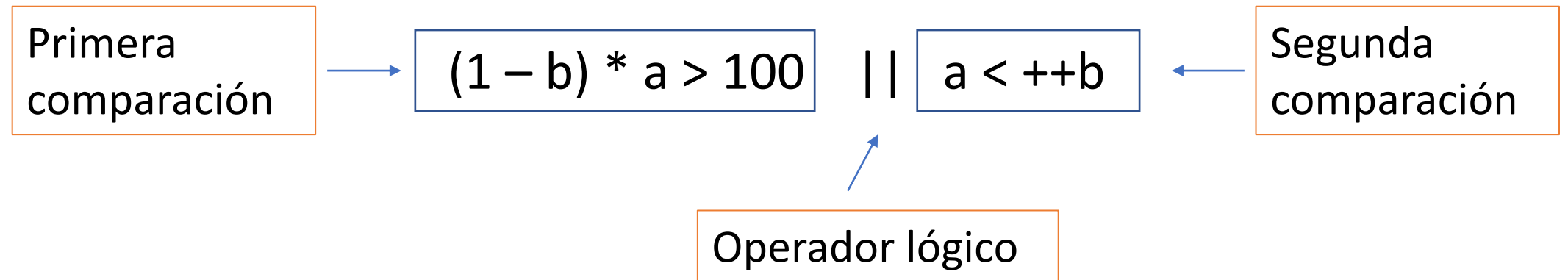
Tabla no

	T	F
!	F	T

Expresiones lógicas

Una **expresión lógica** consta de las siguientes partes:

- Una primera comparación
- Un operador lógico
- Una segunda comparación



Evaluación de expresiones lógicas

Se evalúa la primera comparación



$2 * c + 10 > 100$

$\&\&$

$a == b$



Si procede, se evalúa la segunda comparación y se obtiene el resultado de la expresión lógica



El resultado puede ser true o false



En función del resultado de la primera comparación y del operador lógico puede ser que ya se sepa el resultado de la expresión lógica

Ejemplos. Sean las variables: `int a = 2, b = 3, c = 4;`

`a > 3 && a == 2`

`2 > 3 && 2 == 2`

`false && 2 == 2`

false

`a == 2 && a > 3`

`2 == 2 && 2 > 3`

`true && 2 > 3`

`true && false`

false

`a > 3 & a == 2`

`2 > 3 & 2 == 2`

`false & 2 == 2`

`false & true`

false

`b % 2 == 1 || c > 4`

`3 % 2 == 1 || 4 > 4`

`1 == 1 || 4 > 4`

`true || 4 > 4`

true

`c > 4 || b % 2 == 1`

`4 > 4 || 3 % 2 == 1`

`false || 3 % 2 == 1`

`false || 1 == 1`

`false || true`

true

`b % 2 == 1 | c > 4`

`3 % 2 == 1 | 4 > 4`

`1 == 1 | 4 > 4`

`true | 4 > 4`

`true | false`

true



Planteamiento de expresiones lógicas

“a” es un valor entre 10 y 100, ambos incluidos `10<=a && a<=100`

“a” es par o menor que “b” `a%2 == 1 || a<b`

La diferencia entre la “venta” y el “coste” es superior al doble del “coste” `venta-coste > 2*coste`

La edad de Pedro (“edadPedro”) es mayor que la de Juan (“edadJuan”) y menor que la de Eva (“edadEva”)

`edadPedro > edadJuan && edadPedro < edadEva`

Ejemplo de Expresiones lógicas



//Constantes

```
final int MINIMO, MAXIMO;
```

//Variables

```
double altura = 0;
```

//Programa

```
MINIMO = in.leerInt("ALTURA MINIMA: ");
```

```
MAXIMO = in.leerInt("ALTURA MAXIMA: ");
```

```
altura = in.leerDouble("Altura de la torre: ");
```

```
System.out.println("La altura está entre "  
    + MINIMO + " y " + MAXIMO + ": "  
    + (MINIMO <= altura && altura <= MAXIMO));
```

La expresión lógica tiene que ponerse entre paréntesis

Hágase un programa que lea las constantes enteras mínima y máxima altura. En el programa se lee la altura real de una torre y se comprueba si dicha altura está entre los valores mínimo y máximo.



Salida
consola

ALTURA MINIMA: 50

ALTURA MAXIMA: 80

Altura de la torre: 76,45

La altura está entre 50 y 80: true

Evaluación de múltiples operadores lógicos

En una expresión lógica pueden **aparecer dos o más operadores lógicos**. La forma de evaluar tal operación o expresión sería la siguiente:

ORDEN	OPERADORES
1	()
2	&
3	
4	&&
5	

Ejemplos.

Sean las variables:

int a = 1, b = 2, c = 3;

ASOCIATIVIDAD	1,2,3,4,5	Izquierda a derecha
a-b>0 b==c a>c	a>3 && b == 2 a < c	
1-2>0 2==3 1>3	1>3 && 2 == 2 2 < 3	
-1>0 2==3 1>3	false && 2 == 2 2 < 3	
false 2==3 1>3	false 2 < 3	a > 3 && (b == 2 a < c)
false false 1>3	false true	1 > 3 && (2 == 2 2 < 3)
false 1>3	true	1 > 3 && (true 2 < 3)
false false		1 > 3 && true
false		false && true
		false



Planteamiento de expresiones lógicas

“a” es una vocal en mayúsculas:

`a=='A' || a=='E' || a=='I' || a=='O' || a=='U'`

“letra” es una letra en minúscula sin acentos ni diéresis:

`'A' <= letra && letra <= 'Z' || letra == 'Ñ'`

“peso” es un real con un máximo de dos decimales:

`(int) (peso*100)/100.0 == peso`

“n” es un entero entre -10 y 10 distinto de 0

`-10 <= n && n <= 10 && n != 0`

Planteamiento de expresiones lógicas (II)

**“a” es un valor distinto de 0
entre -1 y 1, ambos incluidos**

$a \neq 0 \ \&\& \ -1 \leq a \ \&\& \ a \leq 1$

Hágase un programa que lea un salario y compruebe que está entre 1000 y 5000 o bien es múltiplo de 100

```
//Variables
int salario = 0;

//Programa
salario = in.leerInt("Salario: ");

System.out.println(1000 < salario &&
salario < 5000 || salario % 100 == 0);
```

Salario: 80500
true

Salida
consola



Validación de datos de entrada numéricos



En la librería “in” los métodos “leerXXX” permiten la validación de los datos de entrada mostrando un mensaje de error estándar o específico

```
// "edad" int  
edad = in.leerInt("Edad: ", v->v>5 && v<=100 );
```

“v” representa el valor de la edad introducida, la cual tiene que ser mayor que 5 y menor o igual que 100



Salida
consola

```
Edad: 4  
Valor incorrecto  
Edad: 100
```

Se puede poner un mensaje de error específico:

```
edad = in.leerInt("Edad: ", v->v>5 && v<=100, "La edad  
tiene que mayor que 5 y menor o igual que 100\n");
```

Salida
consola

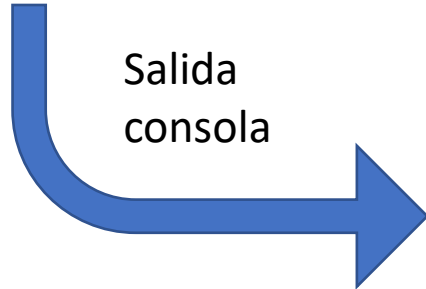


```
Edad: 3  
La edad tiene que mayor que 5 y menor o igual que 100  
Edad: 45
```

Validación de datos de entrada de cadena

En los métodos “leerString” y “leerLine” de la librería “in” se permite validar las cadenas introducidas mostrando un mensaje de error estándar o específico

```
// "estudiante" String  
estudiante = in.leerString("Estudiante (SI/NO): ",  
                           v->v.equals("SI") || v.equals("NO"));
```



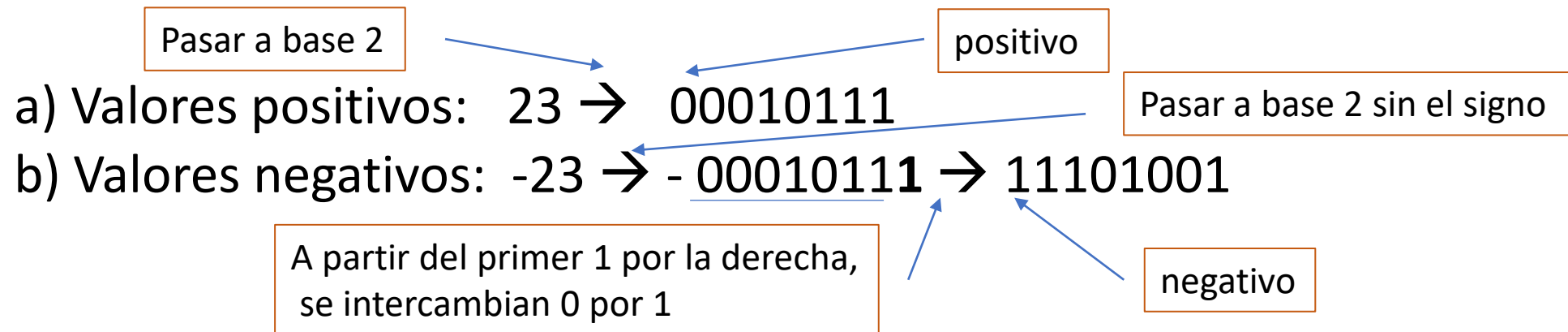
```
Estudiante (SI/NO): si  
Valor incorrecto  
Estudiante (SI/NO): NO
```



Almacenamiento de datos en memoria

Se considera el tipo de datos byte. Con 1 byte (8 bits) se pueden representar 256 valores que van desde el -128 al 127.

Representar en binario un valor de tipo byte.



Obtener un valor de tipo byte a partir de su representación en binario

a) Valores positivos: 00110001 → $1+16+32 = 49$

b) Valores negativos: 10110110 → -01001010 → $-(2+8+64) \rightarrow -74$

Operadores de bits

Se tienen los siguientes operadores de bits:

OPERADOR	USO	OPERACIÓN
&	a & b	Y lógico a nivel de bits
	a b	O lógico a nivel de bits
^	a^b	O lógico exclusivo a nivel de bits
<<	a<<b	Desplazamiento a la izquierda de “a” “b” bits rellenando con ceros por la derecha
>>	a>>b	Desplazamiento a la derecha de “a” “b” bits rellenando con el BIT del signo por la izquierda
>>>	a>>>b	Desplazamiento a la derecha de “a” “b” bits rellenando con ceros por la izquierda
~	~a	Convierte a nivel de bits 0 a 1 y 1 a 0.



Ejemplos de Operadores de bits

Se tienen las siguientes variables: byte a = 5, b = 3, c = -4, d = -6, e = 127

a & b

5 & 3

00000101 & 00000011

00000001

1

a | b

5 | 3

00000101 | 00000011

00000111

7

b ^ c

3 ^ -4

00000011 ^ 1111100

11111111

-1

~a

~5

~00000101

11111010

-7

c >>> 2

-4 >>> 2

11111100 >> 2

00111111

63

c >> 2

-4 >> 2

11111100 >> 2

11111111

-1

d << 1

3 << 1

00000011 << 1

00000110

6

e << 2

127 << 2

01111111 << 2

11111100

-4

Se observa que:

~a = -(a+1)

a << n = a * 2ⁿ (1)

a >> n = a / 2ⁿ

(1) Siempre que no se pierdan bits 1

Evaluación de múltiples operadores de bits

En una expresión de bits pueden **aparecer dos o más operadores de bits**. La forma de evaluar tal operación o expresión sería la siguiente:

ORDEN	OPERADORES
1	()
2	~
3	<< >> >>>
4	&
5	^
6	

ASOCIATIVIDAD

3,4,5,6

Izquierda a derecha

Se tienen las siguientes variables: byte a = 5, b = 3, c = -78

a & (b | 4)

5 & (3 | 4)

00000101 & (00000011 | 00000100)

00000101 & 00000111

00000101

5

c << 1 ^ b

-78 << 1 ^ 3

10110010 << 1 ^ 00000011

01100100 ^ 00000111

01100011

99



Uso de operadores de bits

Sea “a” una variable de tipo byte (8 bits). Los bits se cuentan de derecha a izquierda a partir del 1. El valor **127** se obtiene como **2^8-1**

- **Activar el cuarto bit**

$$a \mid 2^3$$

- **Desactivar el sexto bit**

$$a \& 127 - 2^5$$

- **Comprobar si el tercer bit está activado**

$$(a \& 2^2) == 2^2$$

- **Comprobar si el primer bit está desactivado**

$$(a \& 2^0) == 0$$

$$a \mid 2^5 + 2^3$$

← Activa el 6 y 4 bit

$$(a \& 2^5 + 2^3) == 2^5 + 2^3$$

← Comprobar si el 6 y 4 bit están activados

$$a \& 127 - 2^5 - 2^3 - 2^1$$

← Desactiva el 6, 4 y 2 bit

$$(a \& 2^5 + 2^3) == 0$$

← Comprobar si el 6 y 4 bit están desactivados

Ejemplo de uso de operadores de bits

Hágase un programa que lea un entero de 1 byte “a”, muestre el valor de la expresión “a & 101 << 2” y “(a & 101) << 2”. Compruebe si el tercer y quinto bit están activados, desactive el tercer y quinto bit y muestre el nuevo valor de “a”.

```
//Variables
byte a = 0;
//Programa
a = in.leerByte();

System.out.println(a & 101 << 2);
System.out.println((a & 101) << 2);
System.out.println((a & 16+4) == 16+4);
a = (byte) (a & (127-16-4));
System.out.println(a);
```

```
ESCRIBE UN ENTERO DE 1 BYTE: 52
20
144
true
32
```



Salida
consola

Operadores de asignación (=) y acumulación



OPERADOR	USO	OPERACIÓN
+=	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
&=	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
>>>=	a>>>=b	a = a >>> b

ASOCIATIVIDAD

De derecha a izquierda

Los operadores de acumulación devuelven el valor asignado a la variable.

Sean las variables int a = 7, b = 3, c = -4, d = 5;

a += b	a *= b	a &= b
a += 3	a *= 3	a &= 3
10 (a = 10)	21 (a = 21)	a &= 00 ... 011
		3 (a = 3)

a = b += c *= d
a = b += c *= 5
a = b += -20 (c = -20)
a = -17 (b = -17)
-17 (a = -17)

Instrucciones de
acumulación:



Incrementar en 3 unidades “a”: a+=3;

Duplicar “a”: a*=2;

Asignar múltiples variables: a = b = c = 5;

Ejemplo de Operadores de acumulación

Hágase un programa que lea un número real, lo duplique, se le acumule 100 y se le asigne el resto entre 2,5. Después de cada cálculo se muestra el nuevo valor.

```
//Variables  
double v = 0;
```

```
//Programa
```

```
v = in.leerDouble();  
v*=2;  
System.out.println(v);  
v+=100;  
System.out.println(v);  
v%=2.5;  
System.out.println(v);
```

```
//Variables  
double v = 0;
```

```
//Programa
```

```
v = in.leerDouble();  
v = v*2;  
System.out.println(v);  
v = v+100;  
System.out.println(v);  
v = v%2.5;  
System.out.println(v);
```

```
ESCRIBE UN REAL: 45,7  
91.4  
191.4  
1.40000000000000057
```

Salida
consola



El Operador condicional

OPERADOR	USO	OPERACIÓN
?:	(a)? b : c	Si “a” es true, se devuelve el valor de “b”; si no, el valor de “c”

Ejemplo.

Sean las variables:

int a = 5, b = 6, c = 3;

(a+1>6)? 2*b-1 : c+1

(5+1>6)? 2*6-1 : 3+1

(6>6)? 2*6-1 : 3+1

(false)? 2*6-1 : 3+1

3+1

4

ASOCIATIVIDAD

De izquierda a derecha

(a+1>6)? 2*b-1 : (c>2)? 3*c : 0

(5+1>6)? 2*6-1 : (3>2)? 3*3 : 0

(6>6)? 2*6-1 : (3>2)? 3*3 : 0

(false)? 2*6-1 : (3>2)? 3*3 : 0

(3>2)? 3*3 : 0

(true)? 3*3 : 0

3*3

9

USO: Se utiliza en combinación con el operador de asignación

Instrucción que copia en “a” el doble de “b” si “a” es positivo y si no, “c” menos 2

a = (a>0)? 2*b : c - 2;



Uso del operador condicional

Hágase un programa que lea dos enteros y decida cuál de los dos es mayor

```
//Variables
```

```
int a = 0, b = 0;  
String s = "";
```

```
//Programa
```

```
a = in.leerInt();  
b = in.leerInt();  
s = (a>b) ?
```

```
    "El primero es mayor":
```


```
    (b>a) ?
```

```
        "El segundo es mayor":
```

```
        "Los dos son iguales";
```

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

```
ESCRIBE UN ENTERO: 56  
ESCRIBE UN ENTERO: 11  
El primero es mayor
```



Salida
consola

Orden de prioridad de los operadores

ORDEN	OPERADORES	ASOCIATIVIDAD
1	() [] .	
2	- ~ ! ++ --	
3	new (type)	
4	* / %	De izquierda a derecha
5	+ -	De izquierda a derecha
6	<< >> >>>	De izquierda a derecha
7	< > <= >= instanceof	
8	== !=	
9	&	De izquierda a derecha
10	^	De izquierda a derecha
11		De izquierda a derecha
12	&&	De izquierda a derecha
13		De izquierda a derecha
14	?:	
15	= += -= *= /= %= &= = <<= >>= >>>=	Derecha a izquierda

Operadores pendientes:

[]
.
new
InstanceOf



El operador !



El operador ! tiene máxima prioridad. Si se quiere negar una expresión lógica deberá ponerse el operador ! y entre paréntesis la expresión.

`a >= -100 && a < 100 || a % 2 == 0`

`!(a >= -100 && a < 100 || a % 2 == 0)`

Una alternativa sería negar los operadores de comparación y lógicos según la tabla siguiente:

<	>=
>	<=
==	!=
Solo &&	Solo
&&	()

`a == 1 || a == 10 || a == 100`



`a != 1 && a != 10 && a != 100`

`a >= -100 && a < 100 || a % 2 == 0`



`(a < -100 || a >= 100) && a % 2 != 0`

La librería Math

La librería Math dispone de métodos para calcular potencias, números aleatorios, máximos y mínimos



static double	<code>pow(double a, double b)</code>	Devuelve el valor del primer argumento elevado a la potencia del segundo argumento.
static double	<code>random()</code>	Devuelve un <code>double</code> valor con signo positivo, mayor o igual que <code>0.0</code> y menor que <code>1.0</code> .
static double	<code>max(double a, double b)</code>	Devuelve el mayor de dos <code>double</code> valores.
static float	<code>max(float a, float b)</code>	Devuelve el mayor de dos <code>float</code> valores.
static int	<code>max(int a, int b)</code>	Devuelve el mayor de dos <code>int</code> valores.
static long	<code>max(long a, long b)</code>	Devuelve el mayor de dos <code>long</code> valores.
static double	<code>min(double a, double b)</code>	Devuelve el menor de dos <code>double</code> valores.
static float	<code>min(float a, float b)</code>	Devuelve el menor de dos <code>float</code> valores.
static int	<code>min(int a, int b)</code>	Devuelve el menor de dos <code>int</code> valores.
static long	<code>min(long a, long b)</code>	Devuelve el menor de dos <code>long</code> valores.

Ejemplos. Sean las variables:

```
int a = 7, b = 11, c = 20;  
double d = 0;
```

```
a = (int) Math.pow (b,2); // a = 112  
a = Math.max (b,c); // a = 20  
b = Math.min (a, c); // b = 7
```

Número aleatorio entre 0 y 9,
`c = (int) (Math.round()* 10);`

Número aleatorio entre 6 y 20
`c = (int) (Math.round()* 15) +6;`

Número aleatorio entre -2 y 10
`c = (int) (Math.round()* 13) - 2;`

Número aleatorio entre 2.3 y 8.5
`c = (int) (Math.round()*63)/10.0 +2.3;`

Ejemplo de Uso de la librería Math

Hágase un programa que genere un número aleatorio entero entre -10 y 10; otro entero, entre 0 y 10 y un número real aleatorio entre -10.0 y 10.0. Se muestran los tres valores y el mayor de los tres.

```
//Variables  
int a = 0, b = 0;  
double c = 0, mayor = 0;
```

```
//Programa  
a = (int) (Math.random() * 21) - 10;  
b = (int) (Math.random() * 11);  
c = (int) (Math.random() * 4001) / 100.0 - 20;
```

```
System.out.println(a + " " + b + " " + c);  
mayor = Math.max(a, b);  
mayor = Math.max(mayor, c);
```

```
System.out.println("El mayor es " + mayor);
```

-8 7 -9.22
El mayor es 7.0

Salida
consola



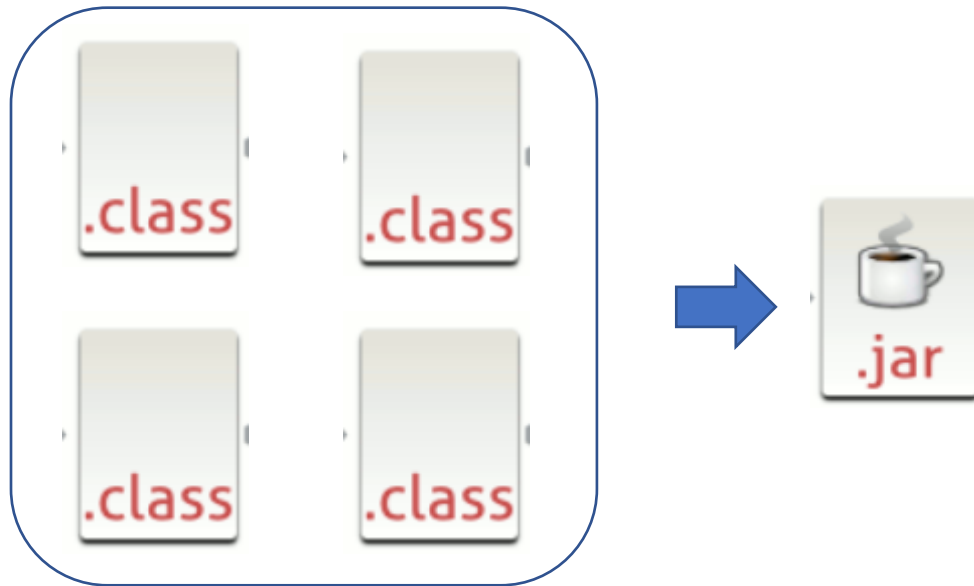
Empaquetar y distribuir

Objetivos

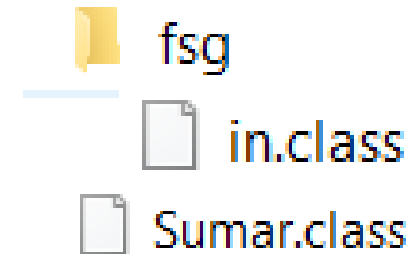
- 1) Empaquetar una aplicación o librería con el JDK
- 2) Descargarse e instalar aplicación launch4j
- 3) Generar una ejecutable “pinchable”

Empaquetar una aplicación o librería

Empaquetar una aplicación consiste en comprimir todos los archivos .class en un solo archivo .jar



- Los .class pueden estar en distintos directorios

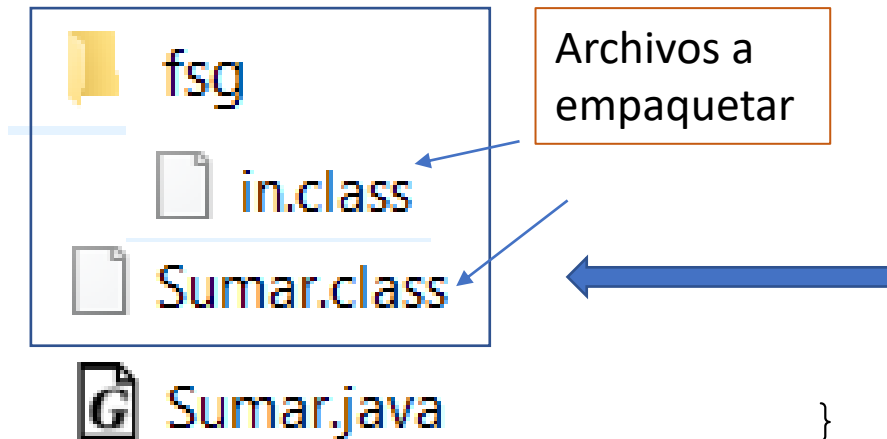


- El jar obtenido será ejecutable si alguno de los .class tiene el método “main”; si no, será una librería

Empaquetar una aplicación con el JDK

Hágase un programa que lea dos enteros y muestre su suma. Previamente se limpiará la consola y al final se detendrá el control.

```
import fsg.in;  
class Sumar  
{  
    public static void main(String[] args)  
    {  
        //Variables  
        int a = 0, b = 0;  
  
        //Programa  
        in.cls();  
        a = in.leerInt();  
        b = in.leerInt();  
        System.out.println("Suma: "+(a+b));  
        in.detener();  
    }  
}
```



Empaquetar una aplicación con el JDK

```
Seleccíon Símboí del sistema  
Microsoft Windows [Versíon 10.0.19041.1052]  
(c) Microsoft Corporation. Todos los derechos reservados.  
  
C:\Users\luiss>cd C:\D\Z\TEORIA\JAVA\POWER POINT\Practicas Tema 01  
  
C:\D\Z\TEORIA\JAVA\POWER POINT\Practicas Tema 01>jar cvfe Sumar.jar Sumar Sumar.class fsg/in.class  
manifiesto agregado  
agregando: Sumar.class(entrada = 1087) (salida = 610)(desinflado 43%)  
agregando: fsg/in.class(entrada = 12019) (salida = 4037)(desinflado 66%)  
  
C:\D\Z\TEORIA\JAVA\POWER POINT\Practicas Tema 01>
```

Directorio por defecto del cmd

Con el comando "cd" se cambia al directorio donde están los .class a empaquetar

Comando del JDK para crear .jar

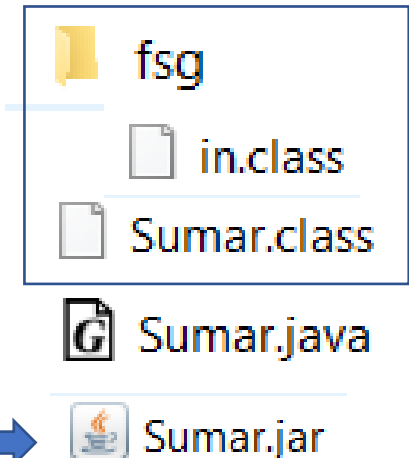
Opciones del comando jar

Nombre del archivo .jar

Nombre de la clase que contiene el main

Archivos .class. a empaque -tar

SE PUEDE PONER *.class POR DIRECTORIO



Ejecutar una aplicación .jar

```
Símbolo del sistema
Microsoft Windows [Versión 10.0.19041.1052]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\luiss>cd C:\D\Z\TEORIA\JAVA\POWER POINT\Practicas Tema 01

C:\D\Z\TEORIA\JAVA\POWER POINT\Practicas Tema 01>jar cvfe Sumar.jar Sumar Sumar.class fsg/in.class
manifiesto agregado
agregando: Sumar.class(entrada = 1087) (salida = 610)(desinflado 43%)
agregando: fsg/in.class(entrada = 12019) (salida = 4037)(desinflado 66%)

C:\D\Z\TEORIA\JAVA\POWER POINT\Practicas Tema 01>java -jar Sumar.jar
```

Directorio donde está el archivo .jar a ejecutar

Comando del JDK para ejecutar aplicaciones

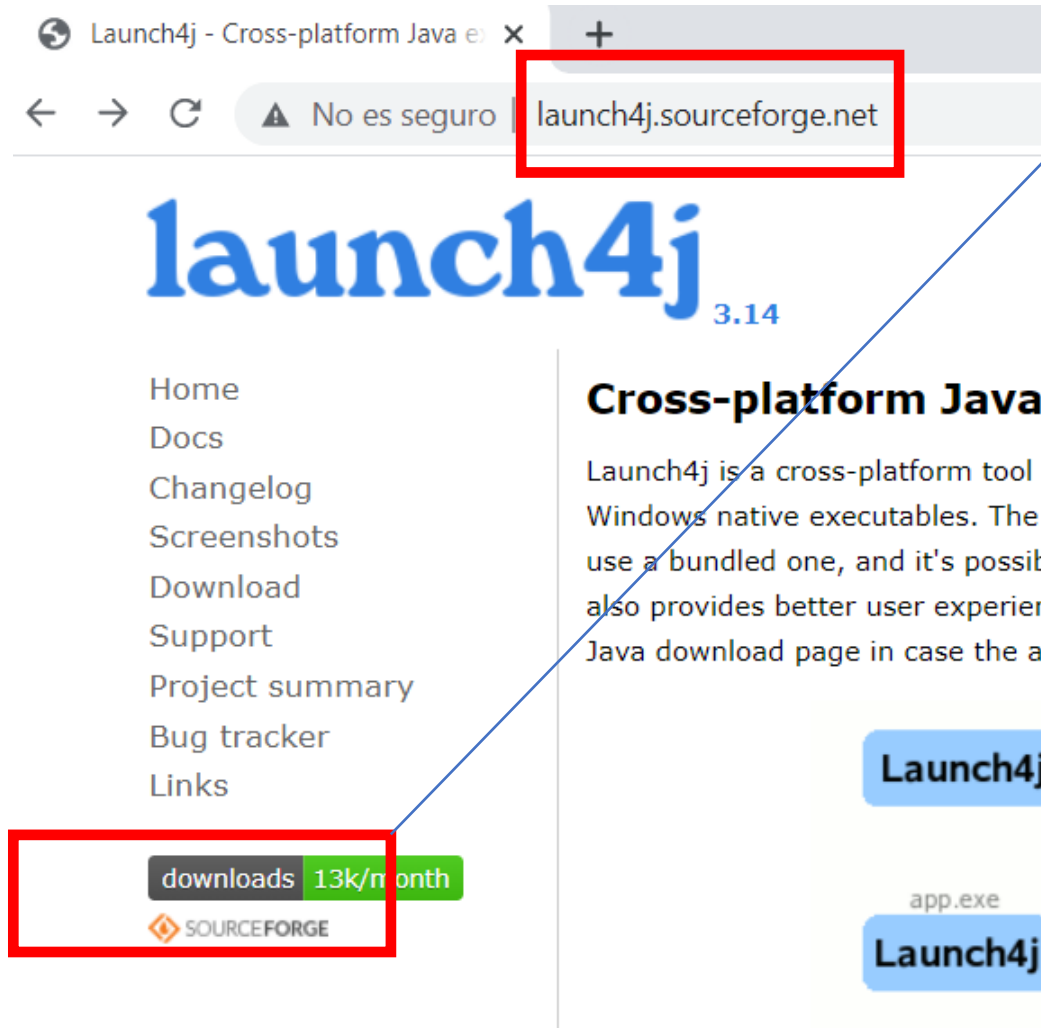
Opción del comando para indicar que se va a ejecutar un .jar

Nombre del .jar a ejecutar

Se inicia la aplicación limpiando la pantalla

```
Símbolo del sistema - j...
ESCRIBE UN ENTERO:
```

Aplicación launch4j



Launch4j - Cross-platform Java executable wrapper

launch4j.sourceforge.net

launch4j 3.14

- Home
- Docs
- Changelog
- Screenshots
- Download
- Support
- Project summary
- Bug tracker
- Links

Cross-platform Java

Launch4j is a cross-platform tool that allows you to create Windows native executables. The use a bundled one, and it's possible also provides better user experience. Java download page in case the app is not installed.

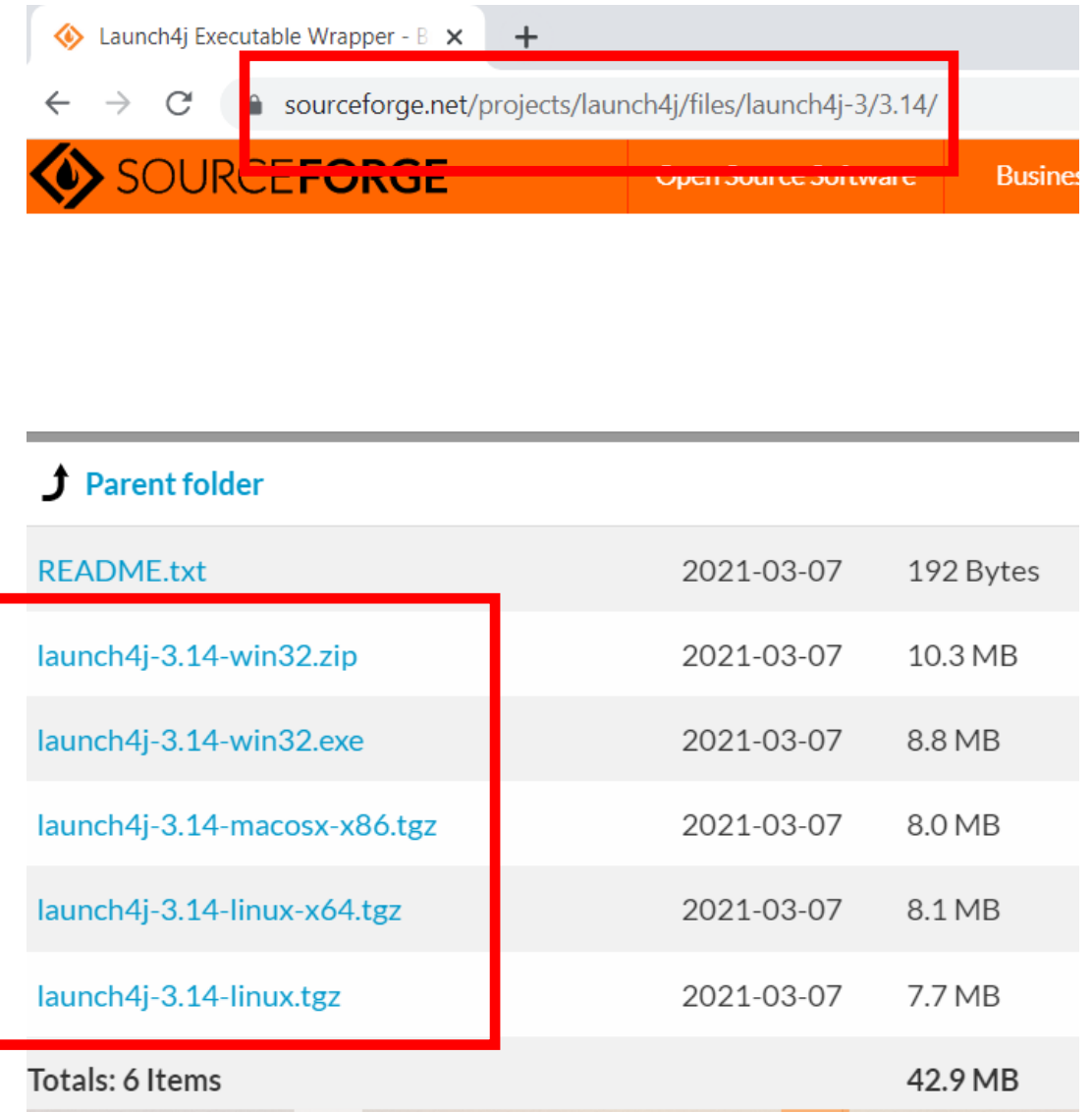
Launch4j

app.exe

Launch4j

downloads 13k/month

SOURCEFORGE



Launch4j Executable Wrapper - B x +

sourceforge.net/projects/launch4j/files/launch4j-3/3.14/

SOURCEFORGE

Parent folder

README.txt	2021-03-07	192 Bytes
launch4j-3.14-win32.zip	2021-03-07	10.3 MB
launch4j-3.14-win32.exe	2021-03-07	8.8 MB
launch4j-3.14-macosx-x86.tgz	2021-03-07	8.0 MB
launch4j-3.14-linux-x64.tgz	2021-03-07	8.1 MB
launch4j-3.14-linux.tgz	2021-03-07	7.7 MB
Totals: 6 Items		42.9 MB

Ejecutable “pinchable”

Al abrir la aplicación “launch4j.exe”
se muestra la siguiente pantalla:

Establecer el nombre del archivo .exe

Buscar el archivo .exe a convertir en .exe

Buscar un icono .ico

The screenshot shows the Launch4j 3.12 - untitled window. The interface includes a menu bar with icons for file operations and settings. Below the menu bar is a tabbed interface with tabs for Basic, Classpath, Header, Single instance, JRE, Set env. variables, Splash, Version Info, and Messages. The Basic tab is active, displaying various configuration fields. Three blue arrows point from the text boxes on the left to specific fields in the Basic tab: the first arrow points to the Output file field, the second to the Jar field, and the third to the Icon field.

Launch4j 3.12 - untitled

Basic Classpath Header Single instance JRE Set env. variables Splash Version Info Messages

* Output file: C:\D\Z\TEORIA\JAVA\POWER POINT\Practicas Tema 01\Sumar.exe

* Jar: C:\D\Z\TEORIA\JAVA\POWER POINT\Practicas Tema 01\Sumar.jar

☐ Don't wrap the jar, launch only

Wrapper manifest:

Icon: C:\D\Z\TEORIA\JAVA\java.ico

Change dir: .

Command line args:

Process priority: ☒ Normal ☐ Idle ☐ High

Options ☒ Stay alive after launching a GUI application ☐ Restart the application after a crash

Java download and support

Error title:

Java download URL: <http://java.com/download>

Support URL:

Log

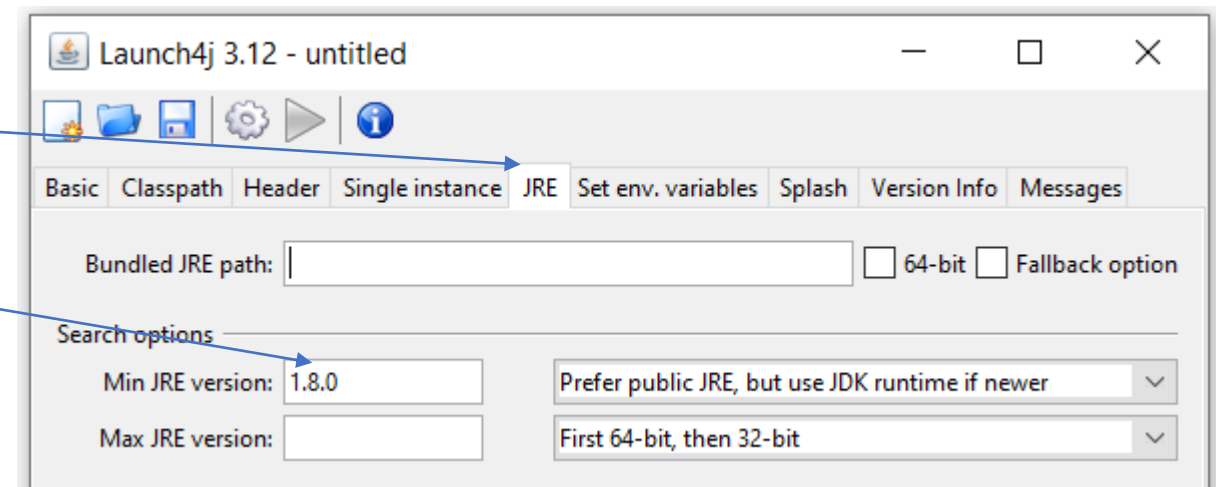
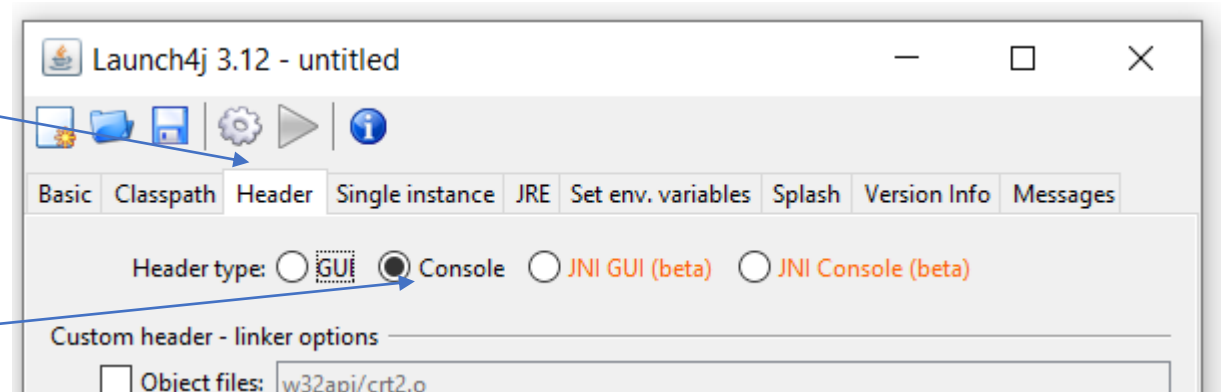
Ejecutable “pinchable” (II)

Se pulsa sobre la pestaña “header”

Se selecciona la opción de consola

Se pulsa sobre la pestaña “JRE”

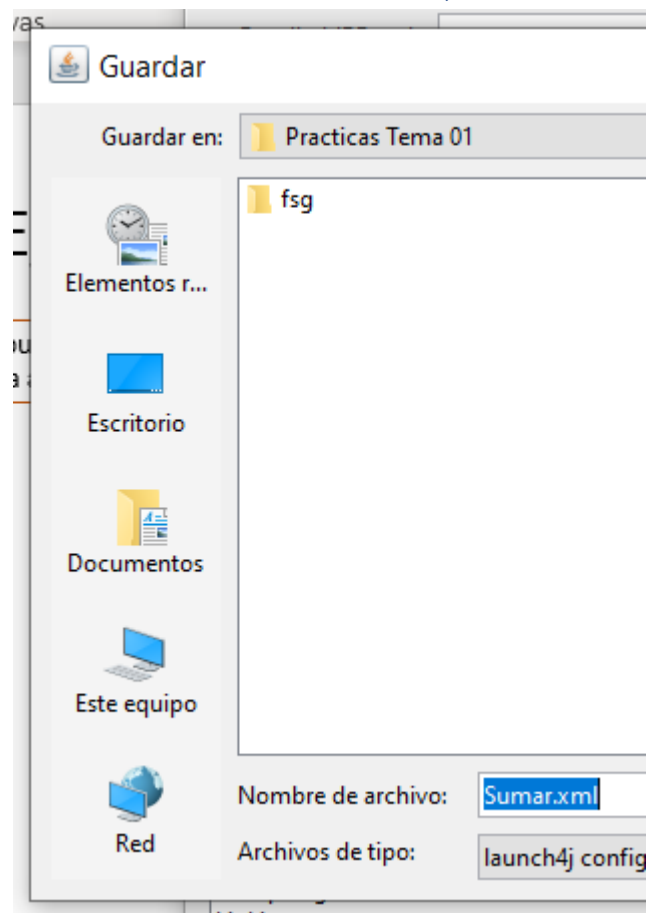
Se escribe la mínima JRE version



Ejecutable “pinchable” (III)



Se pulsa sobre el botón de “guardar”,
para almacenar la configuración



Se pulsa sobre
el botón de
“generar”

