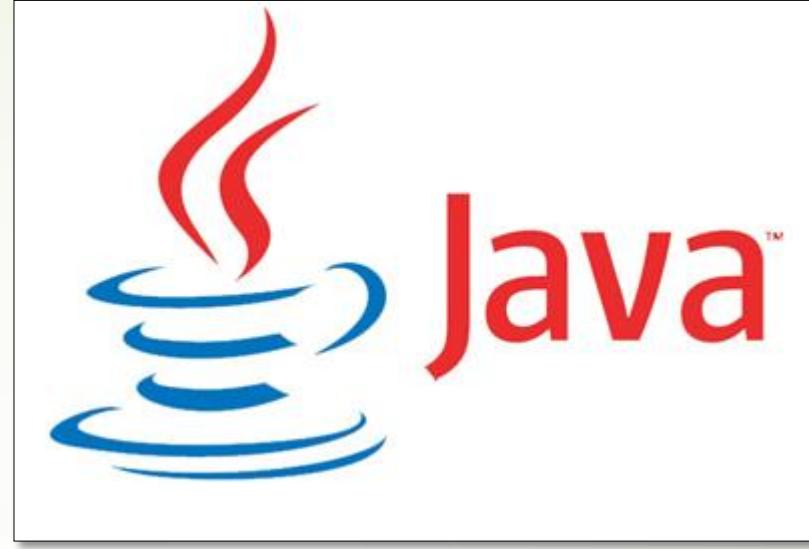




# Desarrollo de Aplicaciones Web y Multiplataforma: Programación

DOCENTE: Daniel López Lozano





# Tema 1.

## Introducción a la Programación en Java

# Índice de contenidos

- ❑ Mi primer programa en Java
  - ✓ Crear, compilar y ejecutar un programa en Java
  - ✓ Partes de un programa en Java
- ❑ Introducción a la programación en Java.
  - ✓ Variables
  - ✓ Expresiones
  - ✓ Operadores
- ❑ Introducción a la entrada y salida de datos.

# Crear, compilar y ejecutar un programa Java

- Paso 1: Creamos en el escritorio una carpeta llamada “misProgramas”.
- Paso 2: Abre el editor de texto “Bloc de notas”
- Paso 3: Crea nuevo fichero llamado “Programa1.java” y guárdalo en la carpeta “misProgramas”
- Paso4: Escribe el siguiente código en el fichero “Programa1.java”:

# Crear, compilar y ejecutar un programa Java

```
//Importamos la libreria java.lang que nos permite imprimir mensajes por pantalla
import java.lang.*;

//Definimos la clase principal de nuestro programa
public class Programa1{

    /*
        Definimos el metodo main. Este es el metodo principal
        de cualquier programa ejecutable en java
    */

    public static void main (String [] args){
        //System.out es una clase de java.lang que permite mostrar mensajes por pantalla
        System.out.println("Hola Mundo");
    }
}
```

Cuidado con las mayúsculas y las minúsculas!!!

- Paso 5: Abre el interprete de comandos de Windows  
inicio->Ejecutar->cmd
- Paso 6: Sitúate en la carpeta misProgramas que hemos creado (utilizando los comandos dir y cd)
- Paso 7: Compila el programa “Programa1.java” que hemos creado. Para ello escribe: javac Programa1.java
- Si no hay ningún error en el código del programa, se producirá la compilación y se generará el fichero intermedio “Programa1.class” con el código ejecutable

- ❑ Paso 8: Comprueba que efectivamente se ha generado el fichero “Programa1.class”.
- ❑ Paso 9: Ejecutamos el programa enviándolo a la máquina virtual de java. Para ello escribimos:  
`java Programa1`
- ❑ Observa que para ejecutar un programa en Java no se escribe la extensión.class

# Partes de un programa en Java

- Java es un lenguaje de programación orientado a objetos (POO).
- Nuestros programas en realidad son clases, una clase se indica mediante la palabra reservada **class**.
- El código fuente se guarda en archivos con el mismo nombre que la clase que contienen y con extensión “.java”

- El compilador genera un **archivo intermedio de clase** (“.class”) por cada una de las clases definidas en el archivo fuente.
- Las clases que se ejecutan de forma autónoma deben contener **forzosamente** el método main.
- Vamos a analizar cada uno de los elementos que componen el siguiente código antes de seguir avanzando:

- Librerías y/o programas importados
- Comentarios
- Bloques de código
- Sentencias
- Metacaracteres
- Variables, constantes, identificadores y expresiones

- Una librería es un conjunto de recursos.
- Si queremos utilizar estos recursos en nuestras aplicaciones, es necesario importar la librería de Java en la que se encuentran.
- Para esto se utiliza la palabra reservada **import** seguido del nombre de la librería a importar.
- Cuando importamos una librería podemos importar sólo una parte de ella indicando que recurso concreto queremos importar, o podemos importarla entera indicándolo con un \*

- ❑ Más adelante veremos como importar nuestros propios programas para reutilizarlos.
- ❑ Los comentarios son líneas de texto insertadas en el programa para documentarlo, facilitar su lectura y comprensión por parte del programador.
- ❑ Los comentarios en Java se indican de 2 formas:
  - ✓ Comentarios de una sola línea:  
//comienzan por 2 barras inclinadas
  - ✓ De varias líneas:  
/\* La primera línea comienza por una barra inclinada y un asterisco y la última termina al revés \*/

- Los bloques de código son la unidad básica de agrupación de instrucciones en un programa. Se forman delimitando por una llave de apertura y una de cierre: {}
- Para mejorar la legibilidad de un código son imprescindibles el uso de comentarios y de las tabulaciones.
- Las instrucciones dentro de un bloque están tabuladas.
- Las tabulaciones ayudan a entender que partes del código pertenecen a un bloque.

# Bloques de código

```
public class Example{  
    // Este bloque es el bloque de definición de La clase Example  
    public static void main(String[] args) {  
        // Este es el bloaque del método main  
        if()...{  
            // Esto es otro bloque  
        }  
    }  
}
```

- Las sentencias son las distintas órdenes que debe ejecutar el programa y terminan siempre con un punto y coma ";" podemos distinguir las siguientes:
  - ✓ E/S: Piden o muestran algún dato por un dispositivo de entrada / salida.
  - ✓ Asignaciones y Expresiones: Las expresiones y sentencias de asignación guardan un valor o el resultado de una operación en un atributo o variable.
  - ✓ Condicionales: Expresan una condición para definir el flujo del programa. if, if- else y switch.
  - ✓ Bucles: Sentencias que se encargan de repetir una o varias sentencias un número determinado de veces. for, while, do-while.

- Los Metacaracteres son una serie de caracteres especiales, que sirven para el control y la significación puntual en las sentencias y los bloques de código:

( ) [ ] { } \ ^ \$ ? ; .

# Entrada-Salida básica

- Para poder realizar una salida de datos por la pantalla es necesario usar la clase System.
- Esta clase tiene el atributo out que hace referencia a la salida estándar (pantalla o monitor).
- Por tanto para mostrar un mensaje por pantalla en primer lugar es necesario importar la librería que contiene a la clase System mediante:

```
import java.lang.*;
```

- Después cuando se deseé mostrar un mensaje por pantalla bastará con escribir la siguiente instrucción:

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

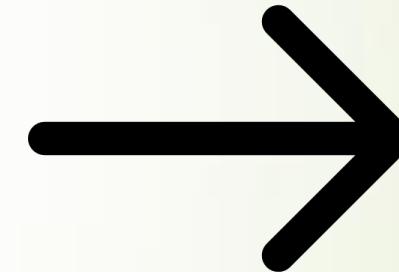
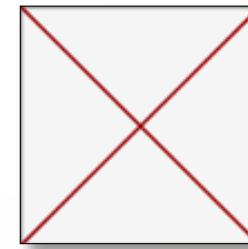
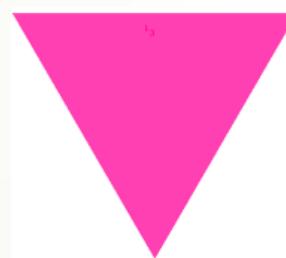
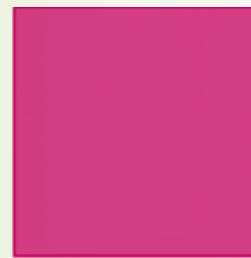
- Donde el mensaje a mostrar puede ser una variable o una cadena de caracteres, Si se trata de una cadena debe ir entre comillas dobles “ ”

- Los caracteres de escape son pequeñas constantes de gran utilizad para formatear las salidas de datos:

```
\t introduce un tabulador: "hola \t mundo" : hola    mundo
-----
\n introduce un salto de línea: "hola \n mundo": hola
                                                       mundo
-----
\" escribir comillas dobles: "hola \"mundo\" ": hola "mundo"
-----
\\ permite escribir una \ : "hola mundo \\": hola mundo \
```

- Escribe una programa que muestre tu nombre por pantalla.
- Modifica el programa anterior para que además se muestre tu dirección y tu número de teléfono. Asegúrate de que los datos se muestran en líneas separadas.
- Modifica el programa anterior para que además muestre por pantalla 3 palabras en inglés junto a su correspondiente traducción al castellano. Las palabras deben estar distribuidas en dos columnas y alineadas a la izquierda (usa el carácter \t).
- Modifica el programa anterior para que además pinte por pantalla una pirámide rellena a base de asteriscos. La base de la pirámide debe estar formada por 9 asteriscos.
- Modifica el programa anterior para que además pinte la pirámide hueca (se debe ver únicamente el contorno hecho con asteriscos).

Crear un programa que dibuje mediante asteriscos  
(u otros caracteres) las siguientes figuras de manera  
aproximada:



- Mostrar siempre texto blanco sobre fondo negro puede resultar muy aburrido.
- El texto que se muestra por pantalla se puede colorear siendo necesario para ello insertar unas secuencias de caracteres.
- Dichas secuencias indican el color con el que se quiere escribir, justo antes del propio texto.

0	1	2	3	4	5	6	7
Negro	Rojo	Verde	Amarillo	Azul	Magenta	Cian	Blanco

```
System.out.println("\u001B[33m\u001B[42mMensaje de color\u001B[0m Ahora normal");
```

- Las tablas completas de colores son:

**NEGRO**

//COLOR DE LETRA

//COLOR DE FONDO

"\u001B[30m"

"\u001B[40m"

**ROJO**

"\u001B[31m"

"\u001B[41m"

**VERDE**

"\u001B[32m"

"\u001B[42m"

**AMARILLO**

"\u001B[33m"

"\u001B[43m"

**AZUL**

"\u001B[34m"

"\u001B[44m"

**MAGENTA**

"\u001B[35m"

"\u001B[45m"

**CIAN**

"\u001B[36m"

"\u001B[46m"

**BLANCO**

"\u001B[37m"

"\u001B[47m"

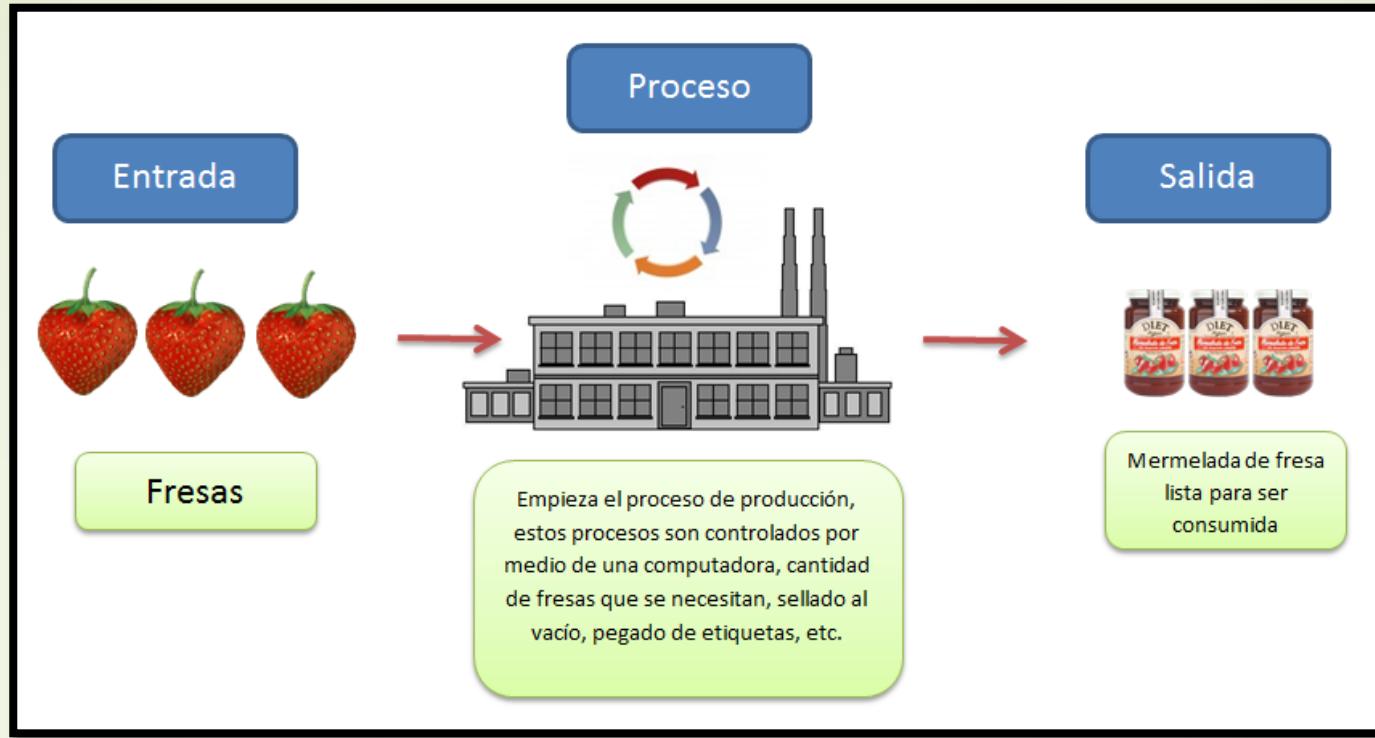
**RESETEAR POR DEFECTO**

"\u001B[0m";

- Java surge como una evolución, un intento de simplificar y mejorar C/C++ con lo cual estos lenguajes coinciden en muchos aspectos básicos:
  - ✓ Variables.
  - ✓ Expresiones.
  - ✓ Operadores.
  - ✓ Constantes.

# Programa = Entrada + Proceso + Salida





- ❑ Una **variable**, es una zona de la memoria del computador donde se puede almacenar un valor que se reserva a lo largo de un programa informático.
- ❑ Algunas de las características de una variable son las siguientes:
  - ✓ Poseen un **identificador** para poder referirse y acceder a ellas.
  - ✓ Su valor puede **cambiarse o modificarse** a lo largo de un programa cuantas veces sea necesario.
  - ✓ Son de un **tipo determinado** que debe indicarse al comenzar a trabajar con ella.

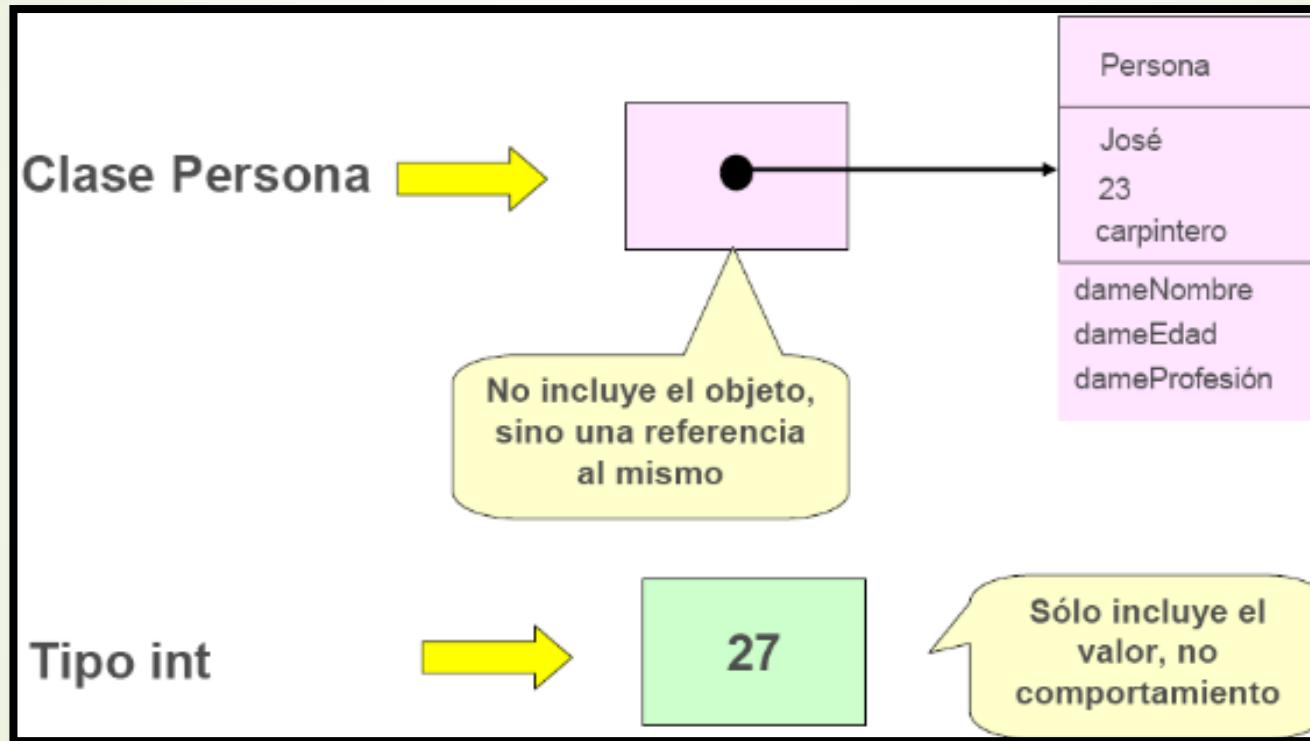
- Conceptualmente las variables podemos pensarlas como cajas de distintos tamaños.



- ❑ El objetivo de las variables es guardar información para que el programa funcione y realice su tarea.
- ❑ ¿Qué variables detectas necesarias en estos casos?
  - ✓ Maquina de jugar a los dardos.
  - ✓ Vitro-cerámica.
  - ✓ Horno o microondas.
  - ✓ Ascensor.
  - ✓ Control de velocidad de un coche.
  - ✓ Libro electrónico.
  - ✓ Reproductor de música.

- Según el tipo de dato que pueden almacenar, podemos distinguir 2 tipos principales de variables:
  - ✓ **Variables de tipos primitivos:** Poseen un valor único y pueden ser entero, decimal, carácter, lógico...
  - ✓ **Variables de tipo objeto:** Se usan para hacer referencia a objetos que tienen que ser de una determinada clase.

# Variables referencias vs básicas



□ **Tipos primitivos:** Son aquellas variables sencillas que contienen tipos de información más habituales: booleanos, caracteres, enteros y reales.

✓ **Enteros:**

- byte(8 bits) valores entre -128 y 127.
- short(16 bits) valores entre -32768 y 32767.
- int (32 bits) valores entre -2147483648 y 2147483647.
- long (64 bits) valores entre  
-9223372036854775808 y 9223372036854775807.

✓ **Reales:**

- float (32 bits) hasta 7 cifras decimales
- double(64 bits) hasta 15 cifras decimales

✓ **Booleanos:**

- boolean (1 bit) true o false

✓ **Caracteres:**

- char (16 bits) (entre comillas simples 'a')

- Cadenas de Texto: Las cadenas de texto no son tipos primitivos, en realidad son objetos.
- Debido a su amplio uso, Java nos permite trabajar con ellas como si fueran tipos primitivos:
- Para diferenciarlas de los caracteres simples van entre comillas dobles.
- “a” es una cadena de texto, ‘a’ es un carácter
- Ejemplos: “hola”, “Esto es una cadena”, “123”, “ ”, “”

❑ ¿Cuál sería el tipo de dato más correcto para los siguientes casos?

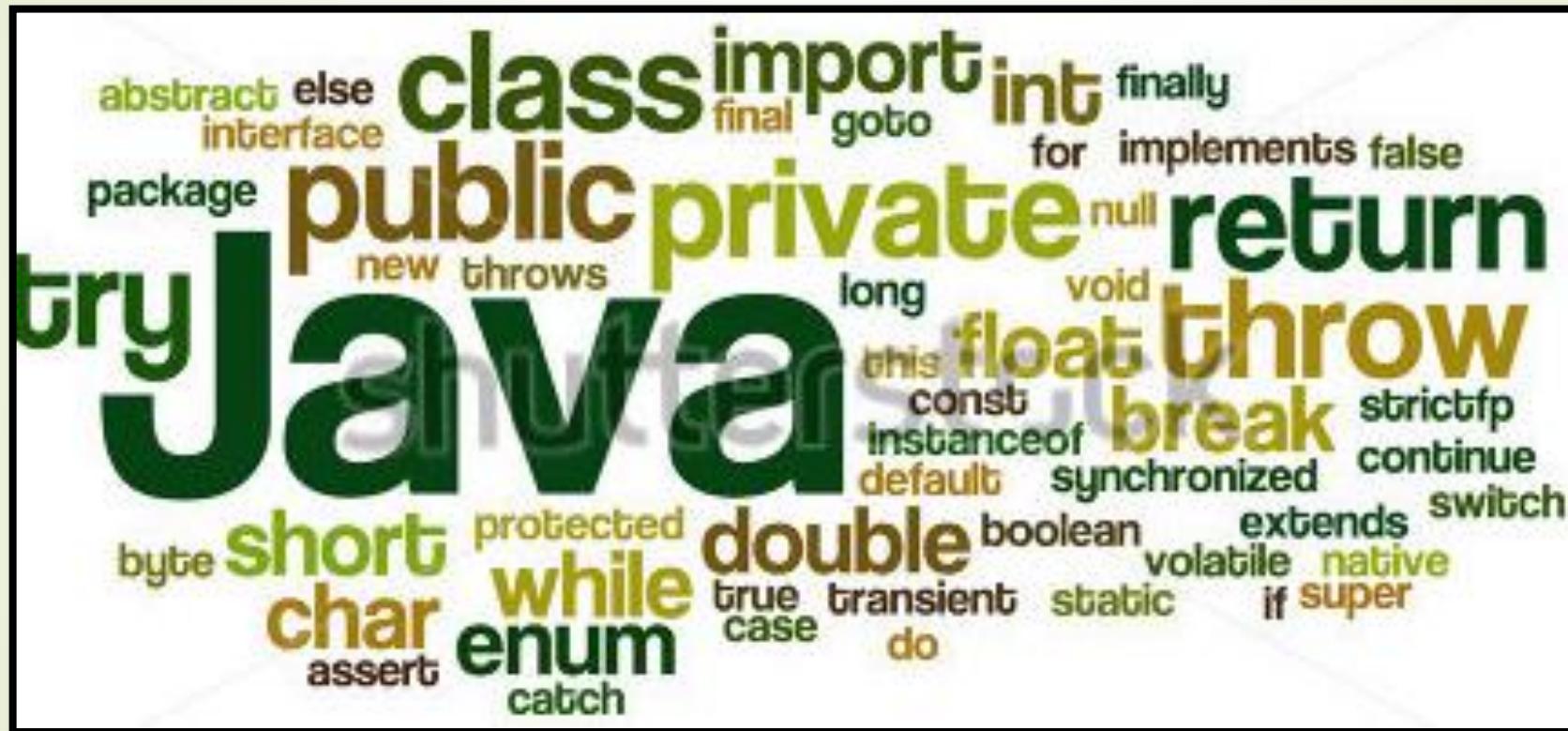
- ✓ Edad de una persona.
- ✓ Sueldo en euros.
- ✓ Número de teléfono.
- ✓ Respuesta a una pregunta con Sí ó No.
- ✓ El valor de PI.
- ✓ Si una persona está casada o soltera.
- ✓ Fecha completa de nacimiento.

- Los identificadores son los nombres que se asigna a las clases, métodos, atributos, objetos, variables, etc, para referirse a ellos dentro del programa.
- Deben respetar las siguientes normas:
  - ✓ No pueden contener ni espacios ni caracteres especiales salvo “\_”.
  - ✓ No pueden empezar por un número.
  - ✓ Java distingue mayúsculas de minúsculas, por lo que “Nombre” y “nombre” son identificadores diferentes.
  - ✓ Se debe evitar el uso de acentos y la letra ñ.

- Además de las normas anteriores, suelen respetarse las siguientes convenciones para asignar identificadores:
  - ✓ El nombre de una clase siempre empieza por mayúscula: **Programa1**
  - ✓ El nombre de un método, variable, atributo etc siempre empieza por minúscula: **main**
  - ✓ Si el identificador está formado por más de un vocablo, a partir del segundo las iniciales deberán ser mayúsculas o guion bajo \_
  - ✓ Los nombres de las clases deberán ser sustantivos, los de los métodos **verbos**.
  - ✓ Los nombres de las **variables** y **atributos** deben expresar con claridad su contenido.

- El nombre de una variable es su identificador, y como tal cumple con las reglas de los identificadores.
- Además en Java existe una serie de palabras reservadas que tienen un significado específico dentro de un programa
- Por lo tanto no se pueden utilizar como nombres de variables. Dichas palabras son:

# Palabras reservadas en Java



¿Cuales son identificadores válidos en Java?

- num80
- 80num
- num\_2
- \_num
- \$pred\$
- Dinero\_de\_la\_caja\_fuerte
- if
- años

# Ejemplo de declaración de variables

```
int puntuacion;  
puntuacion=0;  
  
char letra_DNI;  
letra_DNI='E';  
  
boolean aprobado;  
aprobado=false;  
  
double temperatura=32.56;  
  
String coche="Ferrari";
```

# EJERCICIO

Crear un programa que muestre por pantalla las variables declaradas en la diapositiva anterior

# Ejemplo completo

```
public class Example{
    public static void main(String[] args) {
        boolean miBooleano = false;
        char miCaracter = 'A';
        byte miByte = 0;
        short miEnteroCorto = 1;
        int miEntero = 2;
        Long miEnteroLargo = 3;
        float miFloat = 0.1f;
        double miDouble = 0.2;
        System.out.println("Este es mi booleano: "+ miBooleano);
        System.out.println("Este es mi carácter: "+ miCaracter);
        System.out.println("Este es mi byte:"+ miByte);
        System.out.println("Este es mi entero corto: "+ miEnteroCorto);
        System.out.println("Este es mi entero: "+ miEntero);
        System.out.println("Este es mi entero largo: "+ miEnteroLargo);
        System.out.println("Este es mi float: "+ miFloat);
        System.out.println("Este es mi double: "+ miDouble);
    }
}
```

- Las constantes son un tipo especial de variable, poseen un valor fijo que no se puede cambiar en el transcurso de la ejecución de un programa.
- Estas se declaran mediante el modificador final.

```
final double PI = 3.1416159265;
```

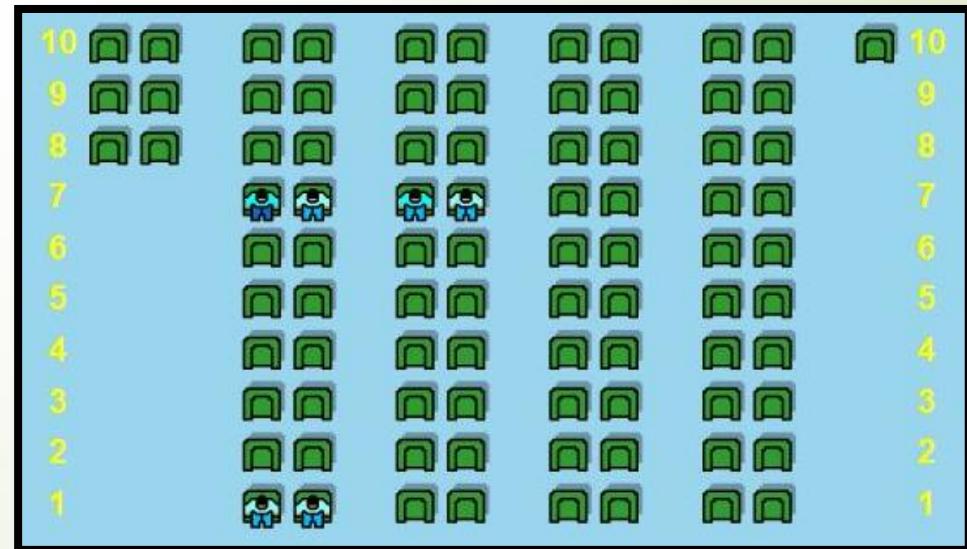
*ó también*

```
final doublé PI;
```

```
PI= 3.1416159265;
```

- Como regla adicional, los identificadores de las constantes suelen declararse en mayúsculas para indicar que lo son.

- Las variables deben utilizarse para guardar información que se ha obtenido en el programa y que puede ser necesario en otro momento.
- El valor constante se refiere a durante la ejecución del programa como contraposición a las variables que si cambian a lo largo del mismo.
- Por ejemplo, aforo máximo de un cine y la asistencia a un sesión de cine.



- Un ejemplo claro de uso de constantes en lugar de variables puede ser para los colores del terminal.
- Son códigos complejos difíciles de recordar, pero que no cambian.

```
final String RESET = "\u001B[0m";
//Colores de Letra
final String NEGRO = "\u001B[30m";
final String ROJO = "\u001B[31m";
final String VERDE = "\u001B[32m";
final String AMARILLO = "\u001B[33m";
final String AZUL = "\u001B[34m";
final String PURPURA = "\u001B[35m";
final String CIAN = "\u001B[36m";
final String BLANCO = "\u001B[37m";
//Colores de fondo
final String NEGRO_BACKGROUND = "\u001B[40m";
final String ROJO_BACKGROUND = "\u001B[41m";
final String VERDE_BACKGROUND = "\u001B[42m";
final String AMARILLO_BACKGROUND = "\u001B[43m";
final String AZUL_BACKGROUND = "\u001B[44m";
final String PURPURA_BACKGROUND = "\u001B[45m";
final String CIAN_BACKGROUND = "\u001B[46m";
final String BLANCO_BACKGROUND = "\u001B[47m";
```

- Declarando una constante le damos a un valor complicado un nombre fácil de recordar.

```
System.out.println(CIAN+VERDE_BACKGROUND+"Mensaje de color"+RESET);
```

- Esto funcionaria con las definiciones de la diapositiva anterior.
- Esto también hace que sea más eficiente que una variable del mismo tipo.

- Las expresiones son fragmentos de código formadas por 2 o más miembros separados entre si por operadores que los evalúan y los relacionan.

- Ejemplo:

$$x = 3 + 2;$$

- El orden es importante, ya que el resultado de una expresión se guarda en la parte izquierda.

$3 + 2 = x; //$ Esto es un error

- Una expresión puede incluir cualquier tipo de operador de los que veremos más adelante.

# Operadores

- ❑ Los operadores aritméticos son operadores binarios (requieren siempre 2 operandos) y corresponden a las operaciones aritméticas habituales:
  - ✓ Suma +
  - ✓ Resta –
  - ✓ Multiplicación \*
  - ✓ División /
  - ✓ Resto de la división %
- ❑ Los operadores de asignación permiten asignar un valor a una variable:
  - ✓ `variable = expresión;`

- En el caso de que no usemos paréntesis en la siguiente tabla se puede observar cual es la precedencia de operadores en Java.
- Teniendo siempre más prioridad de arriba a abajo y de izquierda a derecha.

Tipo de operadores	Operadores
Operadores posfijos	[ ] . ( parámetros) expr++ expr--
Operadores unarios	++expr -expr +expr -expr ~ !
Multiplicación	* / %
Suma	+ -
Comparación	< > <= >=
Igualdad	== !=
AND lógico	&&
OR lógico	
Asignación	= += -= *= /= %=

- El orden en que se realizan las operaciones es fundamental para determinar el resultado de una operación.
- Por ejemplo, el resultado de la operación  $3+2*5$  dependerá de que operación se realice primero.
- El mejor método es indicar en qué orden queremos que se realicen las operaciones. Para esto Java utiliza los paréntesis () .
- En una expresión siempre se evaluará primero los operadores que estén dentro del paréntesis.

# Ejemplos de variables y operadores

```
char letra;
int x=20;
int y;
int suma;
x=x+2;
y=x+56;
suma=23+x+y;
double cantidad,precio,total;
cantidad=100;
precio=5;
total=cantidad*precio;
```

```
x=78;
y=x%2;
z=x%5;
cantidad=x*y-z;
cantidad=x*(y-z);
int m=3,n=2;
double x=7.5,res;
res=(7-n)%2*x+9;
res=(4+n/m)*x;
res=m/6.0*n;

int a=9;
double b=0.5;
res=a+3/a;
res=25/((a-4)*b);
res=a/b*a;
res=a%2-2%a;
```

# Construye expresiones de las siguientes formulas

$$\frac{3x - 1}{x^2}$$

$$\frac{1}{2} + \frac{1}{xy}$$

$$\left(\frac{3 - k}{4}\right)^2$$

$$\frac{9x - (4.5 + y)}{2x}$$

$$3x^2 + 4x^2 :$$

$$9a^2b + ab^2 :$$

$$3x^3yz^2 - 4x^3yz^2$$

$$\frac{3}{2}a^2b + \frac{1}{3}a^2b =$$

$$\frac{x}{3} + \frac{2}{x} - \frac{3x+10}{3x} - \frac{9(x-1)}{3x^2-2x-2} - \frac{1}{x}$$
$$\frac{x-3}{2(x-1)} - \frac{1}{x}$$

# Ejemplo de declaración de variables

```
int xlq3z9ocd = 35.0;
int xlq3z9afd = 12.50;
xlq3p9afd = xlq3z9ocd * xlq3z9afd;
System.out.println(xlq3p9afd);

int a = 35.0;
int b = 12.50;
c = a * b;
System.out.println(c);

int horas = 35.0;
int precio= 12.50;
dinero = horas * precio;
System.out.println(dinero);
```

¿Con que versión os quedáis?

# Operadores de concatenación y aritméticos

```
int year=2018;
System.out.println("hola mundo del "+year+" ¿que tal?");
char puerta='B';
System.out.println("La letra de mi puerta es "+puerta);

int ocupada=3;
System.out.println("La ultima posicion ocupada es la "+ocupada+
                    "y la siguiente libre es "+(ocupada+1));
```

- En algunos casos suele ser necesario convertir un tipo de dato a otro. Este proceso se conoce como conversión de tipos.
- Se debe tener en cuenta el tipo de dato que se va a convertir, ya que si se convierte un dato a otro que tenga una cantidad menor de bits puede haber perdida de información.
- El ejemplo más claro de esto es si se quiere guardar un **número real en una variable entera**. Con lo cual se producirá la perdida de los números decimales.

- Si la conversión se realiza a un tipo de dato con mayor número de bits, la conversión no supone ningún tipo de conflicto y se realiza automáticamente.
- Este tipo de conversión se llama **conversión implícita**.
- Por el contrario si la conversión es a un tipo de dato con menor número de bits que el original, se puede producir perdida de información y hay que indicárselo explícitamente a Java.
- Se la conoce a esto ultimo como **conversión explícita**.

- Para realizar una conversión explícita se tiene que poner el tipo de dato al que se desea convertir entre paréntesis y luego el dato en cuestión:

```
//Un ejemplo de conversión implícita puede ser:  
int numero1 = 32;  
double numero2;  
numero2= numero1; //con esto numero2 pasará a valer 32.0  
  
//Un ejemplo de conversión explícita puede ser:  
double numero1 = 32.17;  
int numero2;  
numero2= (int) numero1; //con esto numero2 pasará a valer 32
```

# Conversiones más especiales

```
char letra='A';
int codigo=letra;
//El número que aparece tiene sentido
//Mirar una tabla ASCII
```

```
int codigo_letra=122;
char caracter=codigo_letra;
//Debería contener la letra z minuscula
```

```
int salto='a'-'A';//Hace conversion implicita
char letra='B'+salto;
```

- La lectura de datos de teclado en un programa en Java no es una operación trivial.
- Para esto son necesarios ciertos conocimientos de programación orientada a objetos que permitan configurar la comunicación con el teclado.
- Para simplificar esta tarea utilizaremos una de las utilidades propias de Java.
- Se trata de la clase Scanner.

- Esta clase se encuentra dentro del paquete “util” de Java, por lo que será necesario importarlo en todos nuestros programas que necesiten usar el teclado:

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

- Para poder trabajar con esta utilidad es necesario inicializarla antes de realizar cualquier lectura de la siguiente forma:

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

- ❑ Donde System.in indica que estamos leyendo del teclado.
- ❑ sc es el nombre que le queremos dar a la conexión que hemos abierto con el teclado.
- ❑ Podemos darle cualquier nombre: sc, lector, teclado...
- ❑ Esta operación sólo hay que realizarla una vez, independientemente de cuantas lecturas queramos hacer.

- La clase Scanner dispone de los siguientes métodos para poder leer datos:

nextBoolean()	//Lee un booleano (True, False)
nextByte() nextShort() nextInt() nextLong()	//Leen enteros
nextFloat() nextDouble()	//Leen decimales
nextLine()	//Lee una linea entera
next()	//Lee una palabra
next().charAt(0)	//Lee un carácter

```
Scanner sc = new Scanner(System.in);
System.out.println("Dime un número");
int num=sc.nextInt();
System.out.println(num); //muestra el número leído por pantalla
```

- Los datos se cogen desde el buffer de teclado.

- ❑ Al introducir un dato numérico de cualquier tipo, se introduce dicho número más el carácter de intro ('\n').
- ❑ Al realizar la lectura, solamente se lee el número y el salto de línea permanece en el buffer de lectura.
- ❑ Esto puede ser problemático si después de leer un número, se lee una cadena de caracteres, ya que esta tomará ese ultimo salto de línea del teclado en lugar de realizar una lectura.
- ❑ Para evitar este problema, basta con realizar una lectura extra para eliminar dicho salto de línea.

# Ejemplo de vaciado del buffer de teclado

```
//inicialización del teclado
Scanner sc = new Scanner(System.in);
//pedimos el número
System.out.println("Dime un número");
int num=sc.nextInt(); //Lectura del número
//pedimos el nombre
System.out.println("Ahora dime un nombre");
sc.nextLine(); //eliminamos lo que pueda haber en el buffer de lectura
String cadena=sc.nextLine();
//mostramos los dos datos leídos
System.out.println("has introducido el número "+num+" y el nombre "+cadena);
```

# Programa completo con E/S

```
import java.util.Scanner;
public class Example{
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Introduce tu nombre completo: ");
        String cadena = scanner.nextLine();
        System.out.println("Has introducido: "+ cadena);
        System.out.print("Introduce tu edad: ");
        int entero = scanner.nextInt();
        System.out.println("Has introducido: "+ entero);
        System.out.print("Introduce tu salario: ");
        double real = scanner.nextDouble();
        System.out.println("Has introducido: "+ real);
    }
}
```

- Operadores incrementales al mismo estilo que los anteriores son incremento (++) y decremento (--) modifican en una unidad la variable a la que se aplica.

`x++` equivale a  $x = x + 1$

`x--` equivale a  $x = x - 1$

- El operador + se utiliza también para unir cadenas de caracteres.
- Para entender mejor observemos los siguientes ejemplos.

- Otros operadores de asignación que incluyen operandos.

<b>Operador</b>	<b>Utilización</b>	<b>Expresión equivalente</b>
<code>+=</code>	<code>op1 += op2</code>	<code>op1 = op1 + op2</code>
<code>-=</code>	<code>op1 -= op2</code>	<code>op1 = op1 - op2</code>
<code>*=</code>	<code>op1 *= op2</code>	<code>op1 = op1 * op2</code>
<code>/=</code>	<code>op1 /= op2</code>	<code>op1 = op1 / op2</code>
<code>%=</code>	<code>op1 %= op2</code>	<code>op1 = op1 % op2</code>

# Ejemplos de operadores abreviados

```
x=0;  
y=4.5;  
z=3*y;  
x=x+1;  
x++;  
x--;  
y=y+5.6;  
y+=5.6  
x=x+y;  
x+=y;  
z=4%3;  
z=x%6;
```

```
dinero+=ventas;  
dinero-=compras;  
dinero-=comisiones;  
dinero*=2;  
dinero/=2;
```

# Bibliografía

- ❑ García de Jalón, j.: “Aprende Java como si estuvieras en primero”. Editorial TECNUN. 2000
- ❑ Holzner, S.: “La biblia de JAVA 2”. Editorial Anaya Multimedia 2000.
- ❑ Moreno Pérez, J.C.: “C.F.G.S Programación” Editorial RA-MA. 2012
- ❑ Wikipedia, la enciclopedia libre.  
<http://es.wikipedia.org/>  
Última visita: Octubre 2018.
- ❑ López, J.C.: “Curso de JAVA  
<http://www.cursojava.com.mx>  
Última visita: Octubre 2018.
- ❑ Documentación oficial Java JSE 8  
<http://docs.oracle.com/javase/8/>  
Última visita: Octubre 2015.
- ❑ Programación en castellano: Java.  
<http://www.programacion.net/java>  
Última visita: Octubre 2015.