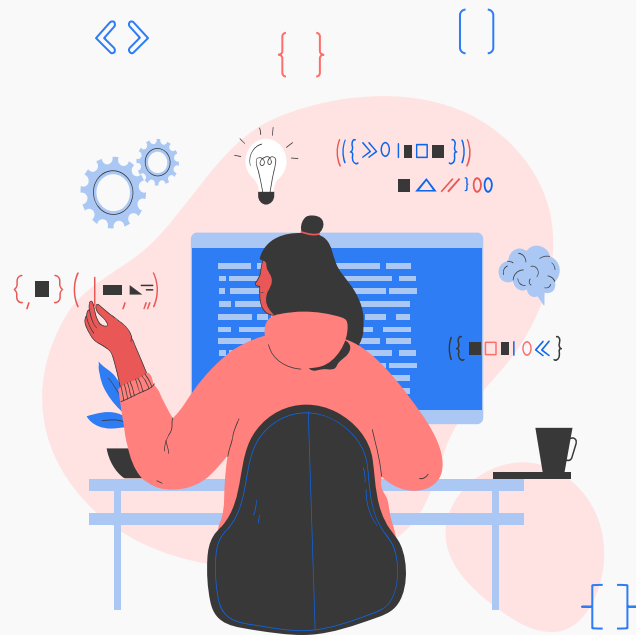


# Programación

## Tema 1 - Introducción a la programación

---

Marina Hurtado Rosales  
marina.hurtado@escuelaartegranada.com



# **Introducción**

---

# Informática

Es el conjunto de conocimientos que permiten procesar la información de manera automática

**Información + automática = informática**

- Dato: cualquier tipo de información como números, palabras, imágenes... que se usa para representar cualquier cosa
- Información: es el resultado de organizar e interpretar los datos. Ej: con una edad, estudios y salario se obtiene la información de un empleado según sus estudios.

- Hardware: son todos los elementos físicos tangibles de un computador
- Software: elementos intangibles de un computador, forman parte de una solución informática.

# Programa informático

Es una serie de instrucciones que indican al computador qué tiene que hacer



Fig.4. Comunicación entre humano y computador.

```
classHolaMundo {  
    publicstaticvoidmain(String[] args{  
        System.out.println("Hola mundo");  
    } // Fin método main  
} // Fin classHolaMundo
```



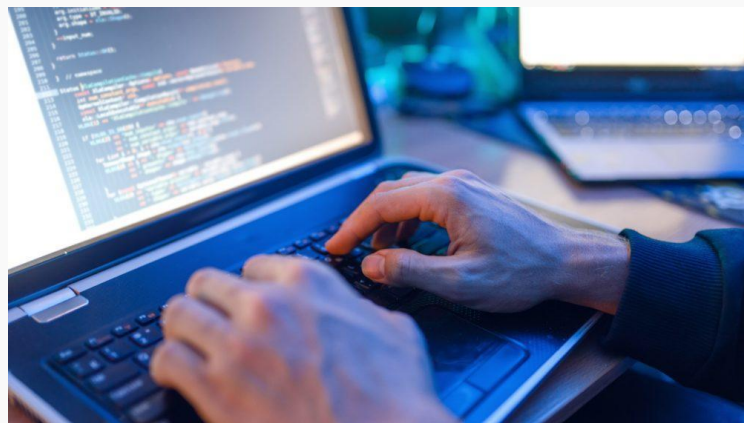
# Lenguajes de programación



Lenguaje natural



Lenguaje de programación



# Lenguajes de programación

Niveles de lenguajes



# Lenguajes de programación

## Lenguaje máquina

```
0110011001100100000000
0000011001100111000110
0100000000001000000000
0000110011001110000000
0011001100111001000000
0000001100110011001100
110010000000000000011001
100111001000000000000011
100110011100100000000000
```

## Lenguaje ensamblador

```
[0x00000000]> pd
0x00000000  90          nop
0x00000001  90          nop
0x00000002  6800009c00  push 0x9c0000 ; 0x009c0000
0x00000007  e8c7ace37b  call 0x7be3acd3
               0x7be3acd3(unk)
0x0000000c  bb04009c00  mov ebx, 0x9c0004
0x00000011  8903        mov [ebx], eax
0x00000013  e81903f47b  call 0x7bf40331
```

## Lenguaje de alto nivel

```
package clases;
import java.util.Scanner;
import clases.Asiento; // Se importa la clase Asiento

class Avion{

    private String matricula, marca, tipo;
    private Asiento asientos[] = new Asiento[80]; // Se crea Array de Objetos de la clase Asiento













    Avion( String matricula, String marca, String tipo){
        setMatricula(matricula);
        setMarca(marca);
        setTipo(tipo);
    }

    /* Metodos Modificadores */
    public void setMatricula(String n){ matricula = n; }
    public void setMarca(String a){ marca = a; }
    public void setTipo(String c){ tipo = c; }
    /* Metodos Accesores */
    public String getMatricula(){ return matricula; }
    public String getMarca(){ return marca; }
    public String getTipo(){ return tipo; }

    public void mostrarAvion(){
        System.out.print("\nMatricula: " + getMatricula() + "\nMarca: " + getMarca() + "\nTipo: " + getTipo() );
    }
}
```

# Lenguajes de programación

Existen 8945 lenguajes de programación! Y aumentan cada día

Sep 2025	Sep 2024	Change	Programming Language		Ratings	Change
1	1			Python	25.98%	+5.81%
2	2			C++	8.80%	-1.94%
3	4	▲		C	8.65%	-0.24%
4	3	▼		Java	8.35%	-1.09%
5	5			C#	6.38%	+0.30%
6	6			JavaScript	3.22%	-0.70%
7	7			Visual Basic	2.84%	+0.14%
8	8			Go	2.32%	-0.03%
9	11	▲		Delphi/Object Pascal	2.26%	+0.49%
10	27	▲		Perl	2.03%	+1.33%
11	9	▼		SQL	1.86%	-0.08%
12	10	▼		Fortran	1.49%	-0.29%

Índice Tiobe Septiembre 2025



# Lenguajes de programación

Existen principalmente dos tipos de lenguajes de programación

[ ]  
Lenguajes compilados:  
Java, C++, C, Fortran...

Lenguajes interpretados:  
Python, Ruby, JavaScript, PHP...

# Sintaxis y semántica

- Semántica: es el significado de lo que está escrito
- Sintaxis: es la forma correcta de escribir y ordenar las palabras

## Ejemplo

SalarioBase (1421,12) + pagaextra (250)= salario\_empleado;

```
double salarioEmpleado  
double pagaExtra = 250;  
double salarioBase = 1421,12;  
salarioEmpleado = salarioBase + pagaExtra;
```

# Paradigmas de la programación

Es el estilo de lucha someter o defenderse de un oponente.  
Tiene técnicas específicas

## Artes marciales

- Kung- Fu
- Judo
- Boxeo
- Sumo
- Capoeira

Objetivo: someter y/o defenderse a un oponente

Es el estilo de programación para resolver problemas. Tiene principios y estructuras específicas

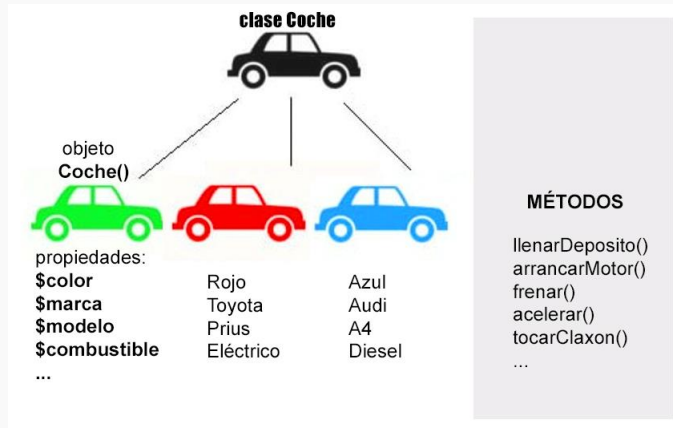
## Paradigmas de programación

- Orientado a objetos
- Funcional
- Secuencial
- Orientado a eventos
- Imperativo

Objetivo: resolver problemas informáticos

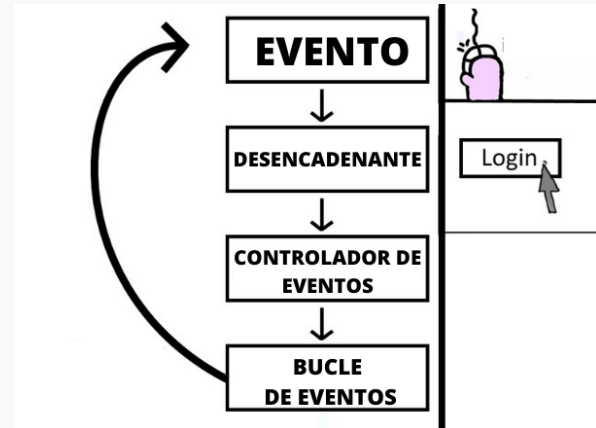
# Paradigmas de la programación

## Paradigma orientado a objetos



Se basa en la reutilización de código  
Ejemplo: Java

## Paradigma orientado a eventos



Ejemplo: Swift

Otros:

- Secuencial
- Funcional
- Multiparadigma

# Dato, algoritmos y programas

Los datos guardan la información que se almacena y procesa

## Ejemplos:

- Nombre
- Edad
- Dirección

## Tipos de datos

- Numéricos
  - Enteros (1, 2, 33, 288, -3, -6) Positivos y negativos
  - Reales (1.5, 3,14, -7,1...) Positivos y negativos
- Carácter
  - Carácter (a, b, 7, -, (, ñ). Todo el alfabeto en todos los idiomas, caracteres especiales, espacios en blanco.
  - Cadena de caracteres (hola, adiós).
- Booleano
  - Verdadero o falso (0 y 1). Evaluación de expresiones

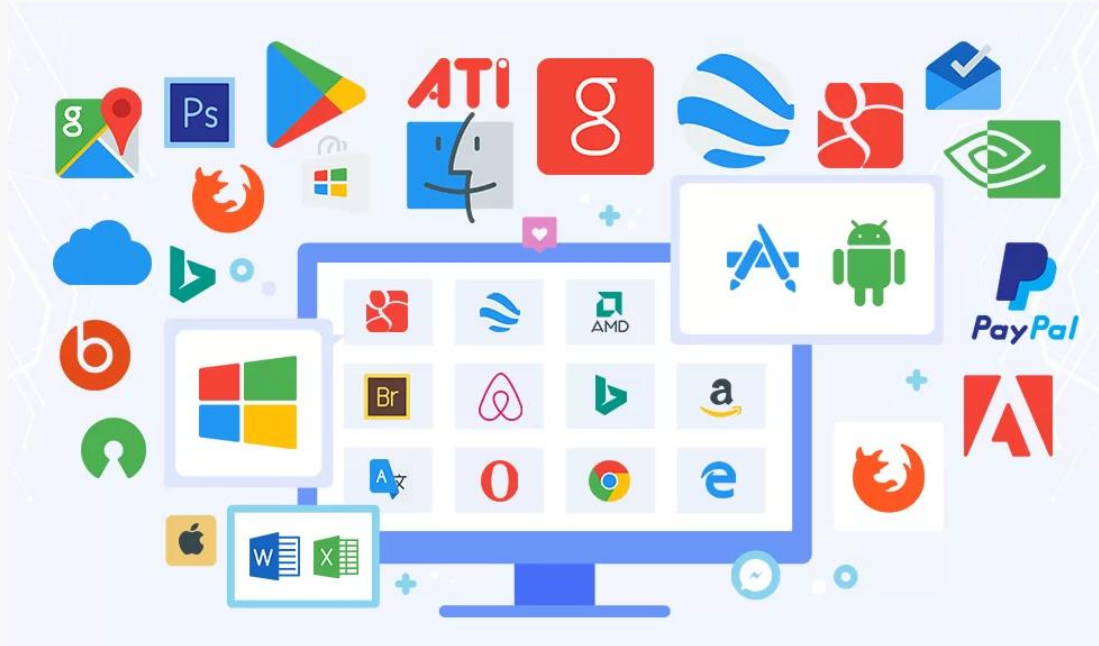
# Dato, algoritmos y programas

Un algoritmo es una secuencia de instrucciones ordenada que resuelve un problema



# Dato, algoritmos y programas

Un programa es el conjunto de instrucciones y algoritmos escritos en un lenguaje de programación



# Ciclo de vida del software

Proceso de creación o desarrollo de un producto software

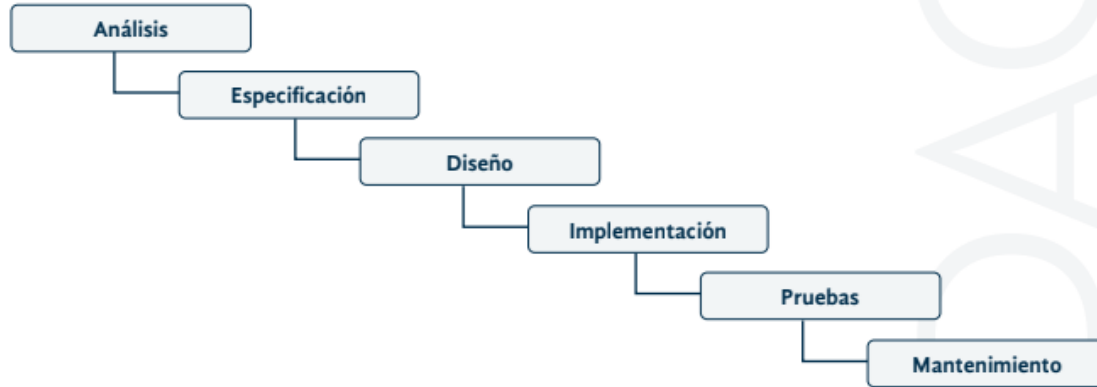


Fig.9. Ciclo de vida del software.

## Ejemplo con una casa

Análisis: cliente te pide qué quiere

Especificación: qué necesitas para construir la casa

Diseño: planos de la casa

Implementación: construcción de la casa

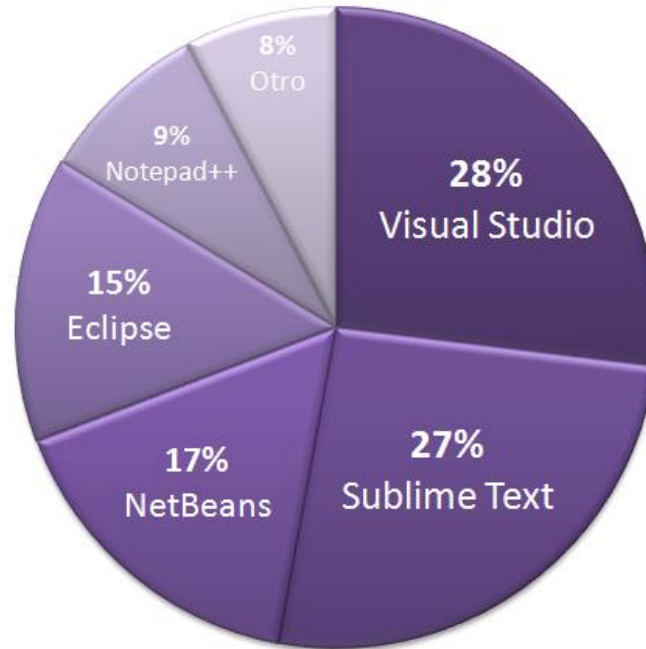
Pruebas: comprobar que todo funciona y no hay fallos

Mantenimiento: revisar y arreglar problemas de la casa



# Entornos de Desarrollo Integrado (IDE)

Son programas que integran herramientas para desarrollar software



# Ejercicio práctico: NetBeans

1. Descargar e instalar JDK 25 del siguiente enlace  
[OpenJDK JDK 25 GA Release](#)
2. Descargar e instalar Netbeans 27 de su página oficial  
[Welcome to Apache NetBeans](#)

Una vez que todo el mundo haya llegado hasta aquí:

1. Creación de un proyecto en NetBeans
2. Compilación de un proyecto en Netbeans

# **Estructura de un programa**

---



# Estructura de un programa



La estructura de un lenguaje de programación es distinta según el lenguaje. No es lo mismo en Java que en C o Python.

## Estructura externa

- Relativo a la organización de los archivos.
- Establece cómo se compone el programa al completo.

## Estructura interna

- Corresponde al código en sí.
- Establece el orden de los distintos bloques de código.
- Incluye las normas de escritura y composición del lenguaje.





# Estructura de un programa

## Estructura externa

- Los programas en Java se denominan Proyectos.
- Todos los proyectos siguen la misma estructura.
- Los nombres de los ficheros se establecen igual en todos los proyectos.





# Estructura de un programa

## Estructura interna

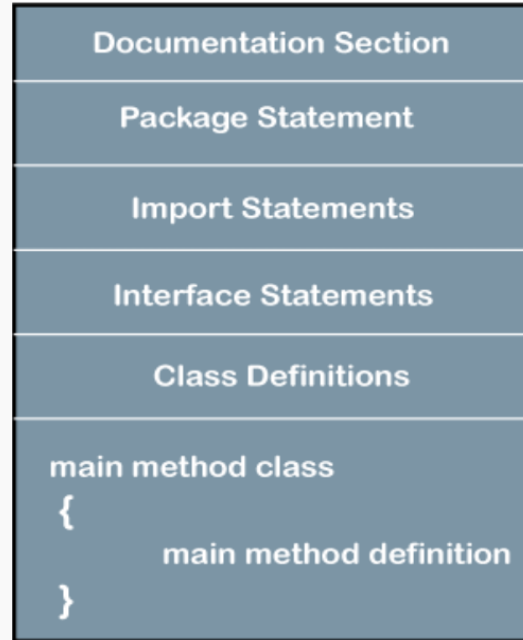
- Abarca tanto como se estructura el propio código, como las reglas de escritura; la sintaxis.
- El elemento más bajo de la estructura son las palabras.
- Las palabras se agrupan en sentencias, las órdenes que rigen el comportamiento del programa. Las sentencias se cierran con “;”.
- Una serie de sentencias que se ejecutan en conjunto para lograr un resultado concreto se denomina bloque de código. Suelen estar acotados por “{}”.



# Estructura y bloques fundamentales

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

# Estructura y bloques fundamentales



Structure of Java Program





# Estructura de un programa

## Estructura interna. Documentación.

- Es una sección opcional pero muy importante, ya que incluye la información básica del programa. Nombre del autor, versión, descripción del programa, etc.
- Esta información es muy útil ya que mejora la lectura y comprensión del programa.
- Para escribir esta información se usan los comentarios, que son sentencias que se ignoran a la hora de compilar.
- Los hay de tres tipos:
  - Comentarios de documentación: `/** */`
  - Comentarios multi-línea: `/* */`
  - Comentarios de una línea: `//`





# Estructura de un programa

## Estructura interna. Import.

- En esta sección se especifican las clases y las funciones predefinidas que queramos añadir a nuestro código. Es opcional.
- Reutilización de código. No vamos a reinventar la rueda.
- Se usa la palabra reservada import.
- Se puede importar una clase en concreto de un paquete o importarlas todas, con \_.
- Ejemplo:

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



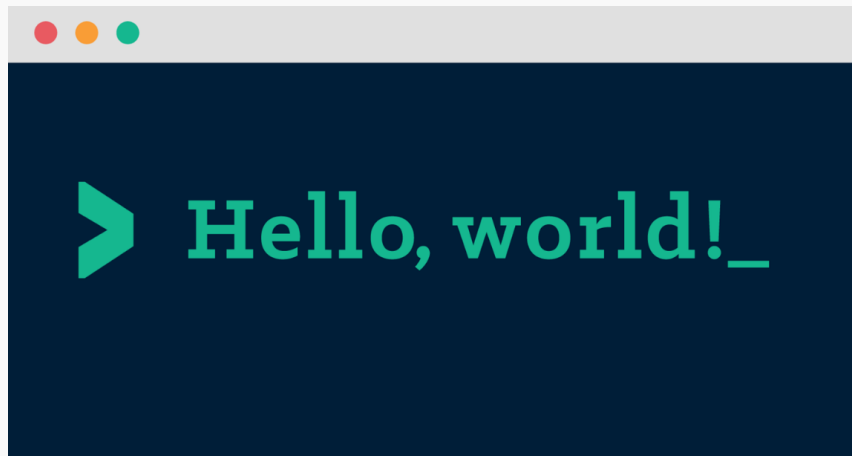


# Estructura de un programa

## Estructura interna. Definiciones de clase.

- En esta sección se define la clase que va a formar parte del programa.
- Es una sección vital, ya que sin una clase no podemos crear el programa.
- Para definir la clase, se usa la palabra reservada class.
- Las clases son unas plantillas que se usan en los programas de Java para agrupar los bloques de código que definamos.
- En cada programa de Java de haber al menos una clase que contenga el método main().





# **Creamos nuestro primer programa**

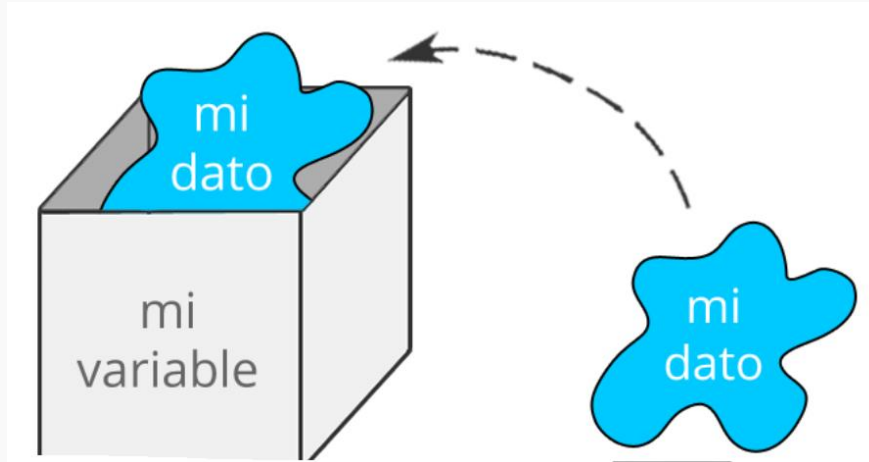
---

# **Conceptos básicos**



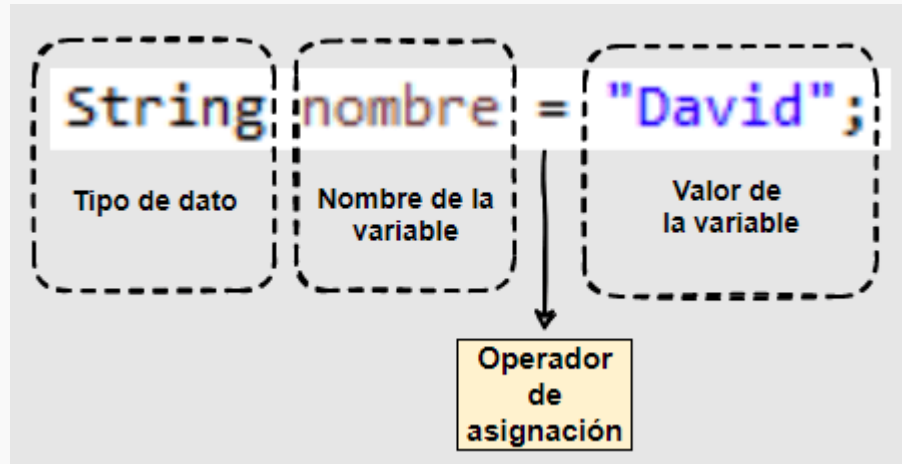
# Variables

- Una variable es un identificador de la zona de memoria en la que se almacena un valor durante la ejecución de un programa.
- En Java, se puede modificar el valor almacenado en tiempo de ejecución.
- De igual forma, son de tipado explícito, es decir, hay que especificar el tipo de dato que van a almacenar.



# Identificadores y variables

Las variables tendrán un valor y un identificador, tendrán una función en concreto



# Identificadores y variables

Los identificadores son los nombres que se da a una variable, función, clase, etc para identificarlos.

Estos identificadores seguirán una norma sintáctica llamada camelCase.

Si un identificador tiene 1 palabra, se escribe **todo** en minúscula.

Si un identificador tiene 2 o más palabras, la **primera en minúscula** y el resto todo **junto** con la **primera letra en mayúscula**.

- nombreEmpresa
- salarioBaseAjustado
- pagos

Estaría mal hacerlo de estas formas:

- Nombre\_empresa
- Salariobaseajustado
- Pagos



# Tipos de datos enteros

Tipo	Bytes ocupados en memoria	Rango de valores
byte	1	$[-128, 127]$
short	2	$[-32768, 32767]$
int	4	$[-2^{31}, 2^{31}-1]$
long	8	$[-2^{63}, 2^{63}-1]$

Por comodidad usaremos el dato int

```
int nombre_variable = 8;
```

```
int nombre_variable1, int nombre_variable = 9;
```

# Tipos de datos reales

Tipo	Bytes ocupados en memoria	Rango de valores	
		En los negativos	En los positivos
float	4	$[-3.4\text{E}^{38}, -1.4\text{E}^{45}]$	$[1.4\text{E}^{-45}, 3.4\text{E}^{48}]$
double	8	$[-1.8\text{E}^{308}, -4.9\text{E}^{324}]$	$[4.9\text{E}^{-324}, 1.48\text{E}^{308}]$

`double nombre_variable = 3.141592;`

`float nombre_variable1, nombre_variable2 = 9;`

# Tipos de datos booleanos

Tipo	Bytes ocupados en memoria	Rango de valores
boolean	1	true / false

`Boolean nombre_variable = true;`

`Boolean nombre_variable = false;`

`Final boolean nombre_variable = true;`

# Tipos de datos alfanuméricos

- **char**: Este tipo de dato representará un único carácter. Se representarán entre comillas simples: 'a', '1'...
- **String**: Este tipo de dato representará varios caracteres juntos, es decir, representará lo que se conoce como una cadena de caracteres. Se representarán entre comillas dobles: "Hola, mundo", "Introduzca su nombre:" ...

Si es solo un único carácter usar **char**, si es una cadena de texto **string**.

```
char nombre_variable = 'H';
```

```
String nombre_variable = "Hola que tal";
```

# Palabras reservadas

- En Java existe una serie de palabras que tienen un significado y un uso específico.
- Debido a esto no se deben utilizar como nombres de variables

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

# Resumen

Variable identificador = valor;

int edadPersona = 35;  
String nombreEmpresa = "EAG";  
double salario = 2500,50;  
Float precioProducto = 4,56;  
Boolean salirMenu = true

# ¿Cuales son identificadores válidos?

- mivariable
- miCumple
- variable1
- 1Cateto
- int
- Matriz\_principal
- #porcentaje#
- raiz\_cuadrada



# Constantes

- Es un tipo especial de variable que posee un **valor fijo** el cual **no** se puede **cambiar**.
- Los identificadores de las constantes se escriben en **mayúsculas**.
- Se declaran mediante el modificador **final**.
- **final double PI = 3.1416159265;**
- Las constantes se inicializan de forma explícita para referenciarlas
- durante la ejecución del mismo







# Expresiones

- Son fragmentos de código formado por 2 o más elementos relacionados por un operador.
- Los más importantes son los **operadores aritméticos**:
  - Suma: +
  - Resta: Multiplicación: \*
  - División: /
  - Resto: %
- **Importante** tener en cuenta el orden de los elementos ya que el valor se guarda en la parte izquierda. **Variable = Expresión**



# Expresiones y operadores aritméticos

Solo se pueden hacer operaciones entre los datos del mismo tipo

Operación	Símbolo	Descripción
Suma	+	Suma dos números
Resta	-	Resta dos números
Producto	*	Multiplica dos números
División	/	Divide con decimales dos números
Resto de la división	%	Obtiene el resto de una división

```
public static void main(String[] args) {  
    int numero1 = 6, numero2 = 3;  
  
    int suma = numero1 + numero2;  
    int resta = numero1 - numero2;  
    int producto = numero1 * numero2;  
    double division = numero1 / numero2;  
    int resto = numero1 % numero2;  
}
```

Los operadores incrementales permiten aumentar (++) o disminuir (--) en 1 un valor numérico.

**++x ó x++ -> x = x + 1**

**--x ó x-- -> x = x - 1**

# Expresiones y operadores aritméticos

Los operadores incrementales permiten aumentar (++) o disminuir (--) en 1 un valor numérico.

[ ]

[ ]

# Operadores lógicos

- **Operador AND**, representado como `&&`.
- **Operador OR**, representado como `||`.
- **Operador NOT**, representado como `!`.

AND	TRUE	FALSE	OR	TRUE	FALSE	NOT	
TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	FALSE
FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE

```
public static void main(String[] args) {  
    boolean booleano1 = true, booleano2 = false;  
  
    boolean resultado1 = booleano1 && booleano2;  
    boolean resultado2 = booleano1 || booleano2;  
    boolean resultado3 = !booleano1;  
    boolean resultado4 = !booleano2;  
}
```

# Operadores relacionales y de asignación

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

Los operadores relacionales disponibles en Java son:

- **Operador igual que**, se representa como ==.
- **Operador distinto**, se representa como !=.
- **Operador mayor que**, se representa como >.
- **Operador menor que**, se representa como <.
- **Operador mayor o igual que**, se representa com
- **Operador menor o igual que**, se representa com



# Expresiones. Prioridad.

- Al igual que en matemáticas, los operadores tienen un tipo de prioridad implícito que indica el orden en el que se ejecuta. Este orden va de arriba a abajo y de izquierda a derecha.
- La prioridad de los operadores es la siguiente:

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



# Comentarios de código

Comentario de una línea: //

Comentario multilínea: /\* comentario \*/

```
public static void main(String[] args) {  
    // Aquí declaramos una variable (Comentario de una línea)  
    int numero = 7;  
  
    /*  
        Ahora vamos a multiplicar el número por 5  
        y mostrarlo por pantalla  
        (Comentario multilínea)  
    */  
    numero *= 5;  
}
```

# Introducir datos desde la consola

Se realiza con un paquete, clase y función específica de Java

## Paquete

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

## EJEMPLO

```
double numero1;  
Scanner leer = new Scanner(System.in);  
numero1 = leer.nextDouble();
```

## Funciones del Scanner

nextDouble() para double

nextInt() para int

nextLine() para Strings





# Entrada/Salida

## Lectura de datos.

- La lectura de datos es una operación sumamente importante que no se puede tomar a la ligera.
- Para realizar la lectura de datos por teclado nos ayudaremos de la clase **Scanner**, propia de Java y que se encuentra dentro del paquete **java.util**.
- Para poder utilizar esta clase, hay que inicializarla primero (solo una vez):

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





# Entrada/Salida

## Lectura de datos.

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

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

```
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
```



# Entrada/Salida

## Lectura de datos.

- Antes de introducir los datos en la variable, éstos se almacenan en un buffer. Esto puede llevar a almacenar datos basura.
- Por eso, entre lectura y lectura es importante vaciar el buffer.

```
//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);
```



# Entrada/Salida

## Formato de Strings

- Se pueden usar combinaciones especiales de caracteres para formatear las salidas de Strings. Se usa la combinación de '`\`' y otro caracter.

Secuencia de escape	Valor
<code>\b</code>	Retroceso o <i>backspace</i> (equivalente a <code>\u0008</code> )
<code>\t</code>	Tabulador (equivalente a <code>\u0009</code> )
<code>\n</code>	Nueva línea (equivalente a <code>\u000A</code> )
<code>\f</code>	Salto de página (equivalente a <code>\u000C</code> )
<code>\r</code>	Retorno de carro (equivalente a <code>\u000D</code> )
<code>\"</code>	Doble comilla (equivalente a <code>\u0022</code> )
<code>\'</code>	Comilla simple (equivalente a <code>\u0027</code> )





# Entrada/Salida

## Formato de Strings

- Por otro lado, Java permite añadir color al fondo del texto y al propio texto que se imprima por pantalla. Esto se logra indicando la secuencia de color antes del propio texto. De la siguiente forma:

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





# Entrada/Salida

## Formato de Strings

- Las secuencias de colores completas son las siguientes:

	//COLOR DE LETRA	//COLOR DE FONDO
NEGRO	"\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";



# Ejercicio práctico: calculadora simple

Crear en Netbeans un programa que actúe como calculadora.

Debe tener dos variables de tipo numérico que serán los números con los que haremos las operaciones (**deben introducirse por consola**) y varias variables que representen el resultado de la **suma, resta, producto, división y resto de la división (%)**.

La calculadora debe mostrar TODOS los resultados

Mostrar por pantalla con 5“`systems.out.println()`” los resultados de las operaciones.

Incluir comentarios en cada línea de código describiendo lo que haces.