PROGRAMACIÓN

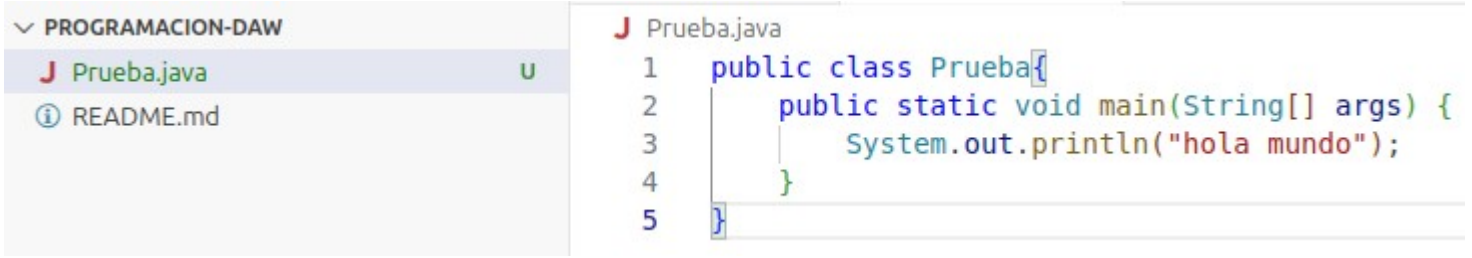
Unidad 2: Fundamentos de programación JAVA

Profesor: Domingo López Oller

ÍNDICE

Palabras reservadas

- Fíjate en el programa Hola Mundo que has creado en la unidad 1. ¿Ves que hay palabras que “pinta” y otras no?



- En JAVA no puedes usar ninguna de las siguientes palabras como nombre de una variable, método o nombre de clase

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

- También hay que añadir los siguientes valores: true, false y null
- Recuerda que todas las instrucciones acaban con ‘;’

Variables

- JAVA es un lenguaje de programación que necesita de variables o “nombres” para poder hacer uso de sus propiedades. Todos ellos tienen un formato de identificador como:
  - Comienzan siempre por un una letra, subrayado o $. Es recomendable usar letra
  - Los siguientes caracteres pueden ser letras, dígitos, subrayado o $
  - Se hace distinción entre mayúsculas y minúsculas. Edad y EdAd son diferentes
  - No hay longitud máxima para definir un identificador
- Ejemplos:

```
public static void main(String[] args) {

    //Identificadores válidos
    String nombre;
    int edad;
    String s;                 //No recomendable, nombre debe ser intuitivo
    double _salario;          //No recomendable, comenzar por letra minuscula
    int $resultado;           //No recomendable, comenzar por letra minuscula
    String Direccion;         //No recomendable, debería empezar en minuscula
    String numerotelefono;    //No recomendable, utilizar camelCase

    //Identificadores no válidos
    String 2apellidos;
    int :idUsuario;
    char .sexo;
    int super;
}
```

Variables

- Además, JAVA es un lenguaje fuertemente tipado, es decir, la variable puede almacenar un entero, una cadena de caracteres, un número real,… pero NO cualquiera de ellos.
- La forma de crear una variable es: tipo id = expresión ‘;’ o varias a la vez separadas por ‘,’. Ej: int x = 10, y = 20, z = 30; es equivalente a int x=10; int y=20; int z=30;
- Estos son los tipos primitivos y los valores que podría tomar:

VARIABLES DE TIPOS PRIMITIVOS.

Nombre	Tipo	Tamaño	Valor por defecto	Forma de inicializar	Rango
Boolean	Lógico	1 bit	False	Boolean a=true	True-false
Char	Carácter	16 bits	Null	Char a='Z'	Unicode
Byte	Numero entero	8 bits	0	Byte a =0	-128 a 127
Short	Numero entero	16 bits	0	Short a =12	-32.768 a 32.767
Int	Numero entero	32 bit	0	Int a= 1250	-2.147.483.648 a 2.147.483.649
Long	Numero entero	64 bits	0	Long a= 125000	-9*10^18 a 9*10^18
Float	Numero real	32 bits	0	Float a =3.1	-3,4*10^38 a 3,4*10^38
Double	Numero real	64 bits	0	Double a = 125.2333	-1,79*10^308 a 1,79*10^308

Práctica: Hay que tener cuidado con las operaciones por el resultado que dan y dónde se guardan
Mira la diferencia entre mostrar (1+2+3)/3 y (1+2+3)/3.0
Puedo forzar el cast colocando el tipo entre paréntesis antes de la expresión. Ej: int a= (int) 2.6;

¿Sabéis qué ocurre si no inicializas una variable? ¿O si le das un valor incorrecto? Prueba int a='a'
Tenéis que entender que TODO se almacena en memoria **RAM** durante la ejecución. Prueba a mostrar a y a[3] cuando la defines como: int[] a={4,0,-1}

Constantes

- Hay una razón por la que las variables se suelen poner en minúsculas. Es de buenas prácticas que en mayúsculas se indiquen las constantes.
- Ejemplos:
  - **final** int MAYORIA_EDAD = 18;
  - final double PI = 3.141592;
- **Importante:** Podrías definir la constante y no darle valor pero una vez se lo asignes ya NO podrás cambiarla.
- Recuerda que lo que hay a la derecha de la asignación (=) es una expresión cuyo valor tiene que coincidir con el tipo de la variable o constante. Ej: final int NUM_ALUMNOS = aulas * 20;

¿Sabéis qué ocurre si no inicializas una variable? ¿O si le das un valor incorrecto? Prueba int a='a'
Tenéis que entender que TODO se almacena en memoria RAM durante la ejecución

Comentarios

- Es importante que vayas comentado tu código, así podrás recordar qué hace tu código y por qué lo pensaste así.
- Hay varias formas de hacer comentarios:
  - **Comentario multilínea:** /* comentarios */
  - **Comentario línea:** // comentario
  - Comentario multilínea para documentación: /** comentarios */ Se emplea con javadoc para obtener una documentación en formato html.

```
// Comentario de una sola línea
```

```
/* Este es un comentario
     tradicional.  Puede
     dividirse en muchas líneas */
```

Ej: javadoc -d doc ruta/*.java

```
/**
 * Programa para imprimir texto.
 * @author ProfesorJava
 * @version 2010
 */
```

Entrada-Salida estándar

- **Salida por pantalla: (System.?)**
  - System.out.print o System.out.println (enlace)
  - ¿Puedo formatear la salida o incluir caracteres especiales? Sí (enlace) Ej: Cambia el -10s a 10s

Le puedes añadir algunos "iconos" como \u264c e incluso colorear el texto o "hacer una tabla" con '-' y '|'

```
System.out.printf("%-10s %10s\n", "Producto", "Precio");
System.out.printf("%-10s %10.2f\n", "Manzana", 1.25);
System.out.printf("%-10s %10.2f\n", "Banana", 0.75);
```

- **Entrada por teclado:**
  - System.in con la clase Scanner (enlace) ¡Ojo con el buffer! (enlace)

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in); //Iniciar la entrada de teclado

    System.out.println("Introduce: Nombre Apellido Edad Profesión");
    String nombre = sc.next();
    String apellido  = sc.next(); //Si se pidiera de uno en uno hay que poner nextLine()

    int edad = sc.nextInt();

    String profesion = sc.next();

    System.out.println("Nombre: " + nombre + ", Apellido: " + apellido+", Edad: " + edad +", Profesión: " + profesion);
    sc.close(); //Cierras el flujo de teclado. Si no lo haces tú ya lo hará el recolector de basura de JAVA
}
```

Este código va a funcionar bien porque se indica todo seguido, veamos el caso de ir pidiendo uno a uno para que entiendas el efecto buffer. Usa sc.nextLine() Necesita hacer sc.nextLine(); o hacer Integer.parseInt(sc.nextLine());

Uso de funciones definidas

- Todos los lenguajes de programación ya tienen una serie de librerías o paquetes con funciones predefinidas. Ej: Math, Swing, Scanner,...
- En el caso de JAVA se produce al principio del fichero con la orden import. Ej: import java.time.LocalDateTime;

```
LocalDateTime hoy = LocalDateTime.now();

System.out.println("Hoy es: " + hoy.getDayOfWeek());   // nombre del día
System.out.println("El día es: " + hoy.getDayOfMonth());
System.out.println("El mes es: " + hoy.getMonth());    // nombre del mes
System.out.println("El año es: " + hoy.getYear());
System.out.println("Hora: " + hoy.getHour() + " Minutos: " + hoy.getMinute());
```

En códigos verás que se usa también java.time.*
El * "inyecta" todas las funciones en tu código JAVA aunque no se usen, hazlo sólo si necesitas bastantes del mismo paquete

- **Práctica:** Veamos la gran ayuda que aporta un entorno de desarrollo si intentaras hacer: System.out.println(**pow(2, 8)**);
  - Aunque haciendo el import static no haría falta hacer Math.pow no es aconsejable por riesgo de ambigüedad.
- **Práctica:** Vamos a hacer ejemplos de uso de la librería Math y a generar número aleatorios (aleatorio=(int)(Math.random() * (max−min+1))+min)
- **Actividad:** Realiza un programa que genera 2 números y nos diga el cociente, la media, la potencia y la raíz cuadrada. Usa tipos adecuados

Uso de funciones definidas

- No reinventes la rueda, si hay funciones definidas por el lenguaje úsalas.
- Si no, te tocará crearlas y utilizarlas donde necesites. Ejemplo: Paquete operaciones aritméticas

```
package utilidades; //carpeta que contiene la clase

public class Matematicas {
    public static int sumar(int a, int b) {
        return a + b;
    }

    public static int multiplicar(int a, int b) {
        return a * b;
    }
}
```

```
import utilidades.Matematicas; //Acceso al paquete

public class Main {
    public static void main(String[] args) {
        int x = Matematicas.sumar(5, 3);
        int y = Matematicas.multiplicar(4, 2);

        System.out.println("Suma: " + x);
        System.out.println("Multiplicación: " + y);
    }
}
```

Incidiremos en esto más adelante cuando veamos el concepto de clase

Operadores

Operadores aritméticos

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
-	operador unario de cambio de signo	`-4`	`-4`
+	Suma	`2.5 + 7.1`	`9.6`
-	Resta	`235.6 - 103.5`	`132.1`
*	Producto	`1.2 * 1.1`	`1.32`
/	División (tanto entera como real)	`0.050 / 0.2` `7 / 2`	`0.25` `3`
%	Resto de la división entera	`20 % 7`	`6`

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo

++	Incremento
	`i++` primero se utiliza la variable y luego se incrementa su valor
	`++i` primero se incrementa el valor de la variable y luego se utiliza
		`4++`
		`a=5;`
		`b=a++;`
		`a=5;`
		`b=++a;`
			`5`
			`a vale 6 y b vale 5`
			`a vale 6 y b vale 6`

--	decremento	`4--`	`3`

Operadores relacionales

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
==	igual que	`7 == 38`	`false`
!=	distinto que	`'a' != 'k'`	`true`
<	menor que	`'G' < 'B'`	`false`
>	mayor que	`'b' > 'a'`	`true`
<=	menor o igual que	`7.5 <= 7.38`	`false`
>=	mayor o igual que	`38 >= 7`	`true`

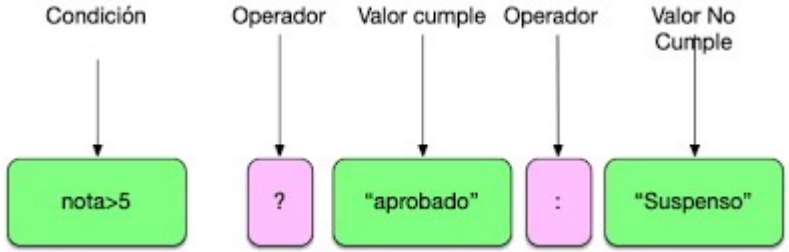
Operadores opera-asigna

Operador de asignación	Expresión abreviada
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

Operadores lógicos

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

Operadores ternario




```
x = a == 10 ? b*2 : a;
```

Condicional

- **IF e IF-ELSE:** Se trata de una instrucción que SI cumple la condición ejecuta su contenido o permite ejecutar una de las dos opciones. Esto se puede encadenar colocando varios else if hasta el último:

```
public static void main(String[] args) {

    int user = 17;

    if (user <= 18) {
        System.out.println("User is 18 or younger");
    }
    else {
        System.out.println("User is older than 18");
    }
}
```

Recuerda que lo que hay dentro del if es un bloque de instrucciones y que la condición puede ser compleja.
Práctica: ¿Cómo sabemos si un número es divisible por 2 y por 3?

```
public static void main(String[] args) {

    int user = 45;

    if (user <= 18) {
        System.out.println("User is 18 or younger");
    }
    else if (user > 18 && user < 40) {
        System.out.println("User is between 19 and 39");
    }
    else if (user == 45 || user == 50) {
        System.out.println("User is either 45 OR 50");
    }
    else {
        System.out.println("User is older than 40");
    }
}
```

Condicional

- La opción de if-else encadenado se puede simplificar cuando hay muchos caso mediante la instrucción **switch**

```
switch (expression) {
    case value1:
     // secuencia de sentencias.
     break;
    case value2:
     // secuencia de sentencias.
     break;
    .
    .
    .
    case valueN :
     // secuencia de sentencias.
     break;
    default:
     // Default secuencia de sentencias.
  }
```

- Ejemplos:

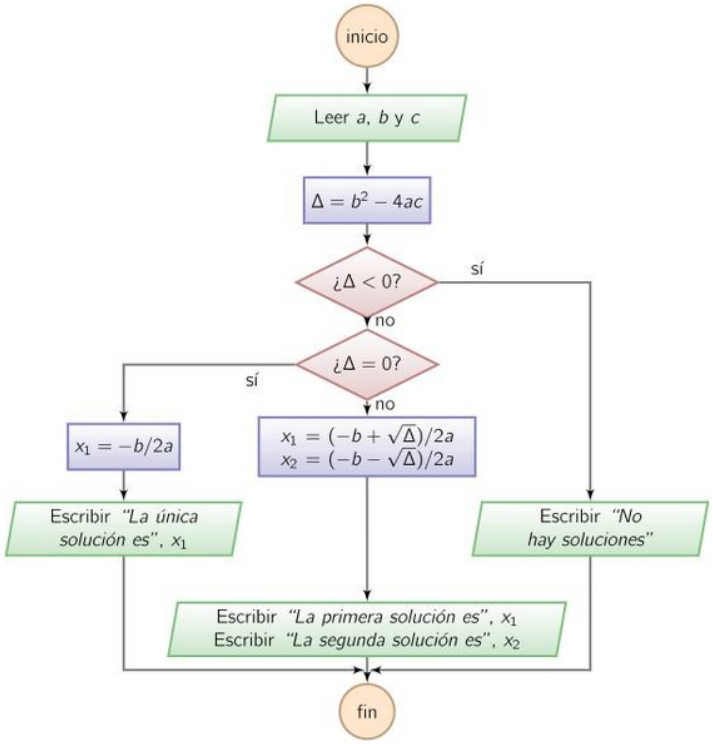
```
int i = 2;
switch(i) {
  case 0:
    System.out.println("i es cero.");
    break;
  case 1:
    System.out.println("i es uno.");
    break;
  case 2:
    System.out.println("i es dos.");
    break;
  case 3:
    System.out.println("i es tres.");
    break;
  default:
    System.out.println("i es mayor a tres.");
}
```

```
int i = 2;
switch(i) {
  case 0:
  case 1:
  case 2:
  case 3:
  case 4:
    System.out.println("i es menor que cinco");
    break;
  case 5:
    System.out.println("i es cinco");
    break;
  case 6:
  case 7:
  case 8:
  case 9:
    System.out.println("i es menor que diez y mayor a cinco");
    break;
  default:
    System.out.println("i es diez o mayor a diez");
}
```

Práctica: Haz un programa al que le pasas un entero y te dice el nombre del mes correspondiente.

Condicional

- **Actividad:** Haz el programa JAVA del siguiente diagrama de flujo



- **Actividad:** Haz un programa que nos pide una nota y nos indica la calificación (sobresaliente, notable, bien, aprobado, suspenso, nota incorrecta). Usa if-else y switch

- **Actividad**: Haz un programa que solicita el día, mes y año y comprueba si es válido. Para el año hay que ver si el 29 es válido por ser bisiesto. Un año es bisiesto si: (anio % 4 == 0 && anio % 100 != 0) || (anio % 400 == 0)

BUCLES

- **Para repetir instrucciones podemos usar:**

Estructura Iterativa

Inicialización

Expresión Booleana

Actualización

```
for (int i=1; i<=12; i++) {
    suma= num + i;
}
```

Instrucciones que se repetirán

No necesita declarar la variable fuera del código y ya se le indica la expresión de paso

Versión for para arrays

```
for (int num : array) {
    ...
}
```

Estructura Iterativa

Expresión Lógica

```
While (i<=n){
    s=s+1;
}
```

Instrucciones que se repetirán

En estos casos hay que tener en cuenta la variable de control (i y nota) y hacer la declaración y la manipulación

Estructura Iterativa

```
do{
    s=s+1;
}
While (nota<=0 || nota>=20)
```

Instrucciones que se repetirán

Expresión Lógica

Evita que un bucle se rompa con la instrucción break, piensa mejor la condición de salida

Tened mucho cuidado con los bucles infinitos por tener mal definida la condición de parada!!

BUCLES

- **Actividades:**

1) Desarrolla un programa que muestre los números entre 50 y 200 que son múltiplos de 2 y 3
2) Desarrolla un programa que calcule el factorial del número introducido. Ej: 4!= 4*3*2*1
3) Desarrolla un programa que muestre la edad máxima y mínima de un grupo de alumnos que se introduzca hasta escribir -1

Hay que tener en mente las situaciones de error. Mira el error que hay con las actividad 2 si introduces una letra.
Hay que analizar la entrada en la medida de lo posible como con la función. Prueba hasNextInt() para tenerlo en cuenta
Hay una metodología dirigida a test (TDD)

BUCLES

- **Actividades:**

4) Continúa el programa anterior para que calcule la suma, media, número de alumnos y cuántos son mayores de edad

5) Desarrolla un programa que trate de adivinar un número entre 1 y 100 generado aleatorio, que vaya indicando si has introducido un número mayor o menor al correcto y el número de intentos.

6) Desarrolla un programa que simule una calculadora según la opción introducida y que pida los operandos hasta que introduces -1

BUCLES

- **Actividades:**

7)

```
Algoritmo MCD
    Leer a, b
    a ← valor absoluto de a
    b ← valor absoluto de b

    Mientras b ≠ 0 hacer
        resto ← a mod b
        a ← b
        b ← resto
    FinMientras

    Escribir "El MCD es: ", a
FinAlgoritmo
```

```
Algoritmo MCM
    Leer a, b
    a ← valor absoluto de a
    b ← valor absoluto de b

    mayor ← máximo(a, b)
    mcm ← mayor

    Mientras (mcm mod a =/0) o (mcm mod b =/0) hacer
        mcm ← mcm + mayor
    FinMientras

    Escribir "El mcm es: ", mcm
FinAlgoritmo
```

Ampliación: Intenta hacer este caso sólo usando funciones de Math

BUCLES ANIDADOS

- **¿Se puede poner un bucle dentro de otro?** SÍ

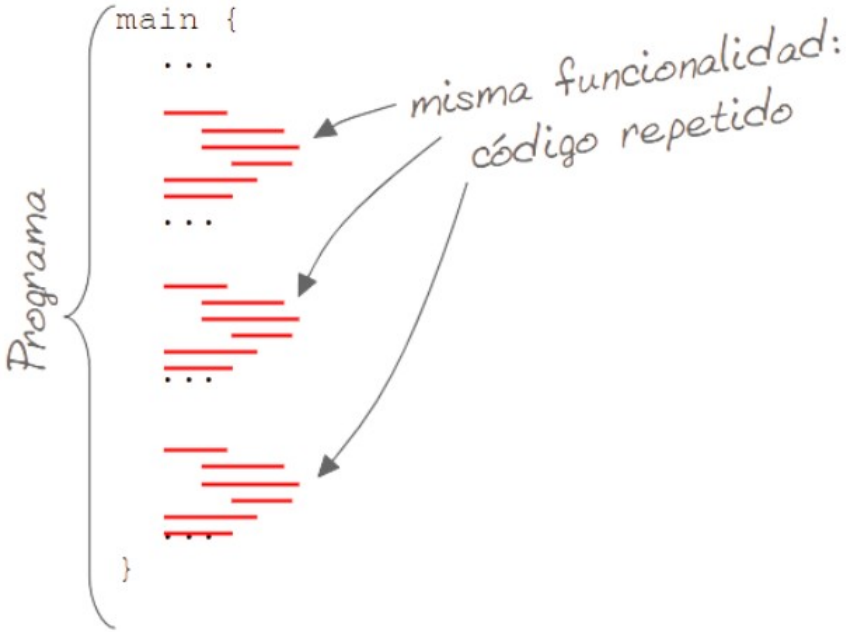
Ejemplo en C++

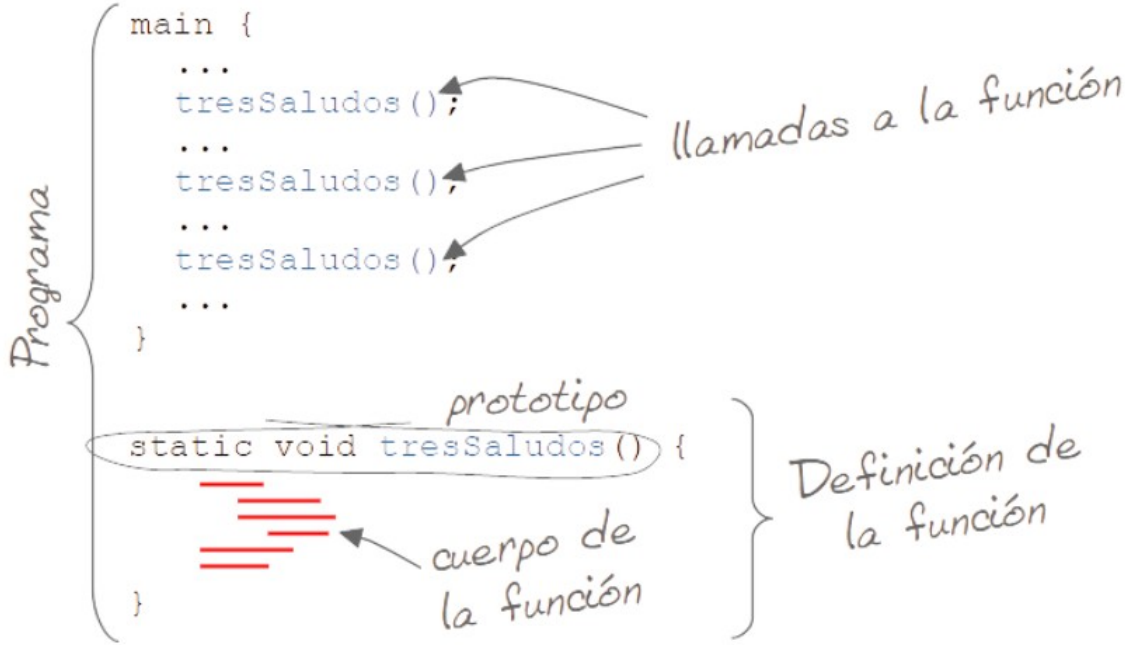
```
for (i=1; i<=4; i++)
{
   for (j=1; j<=i ;j++)
       cout << "*";
   cout << endl;
}
```

- **Práctica:** Haz la variante del ejemplo para que saque el resultado al revés y sólo las filas pares
- **Actividad:** Desarrolla el programa que muestra las tablas de multiplicar del 1 al 10

FUNCIONES

- **No hay que repetir código y hay que organizar el código para que sea legible.**

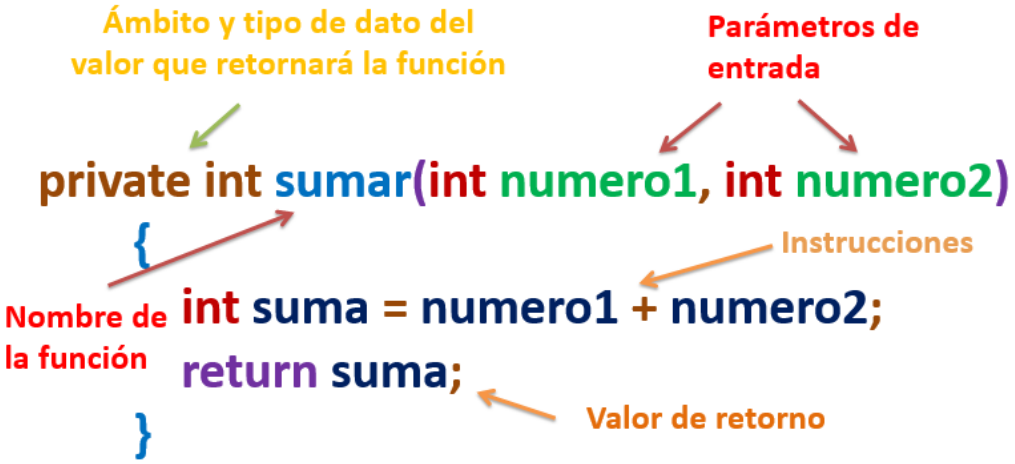


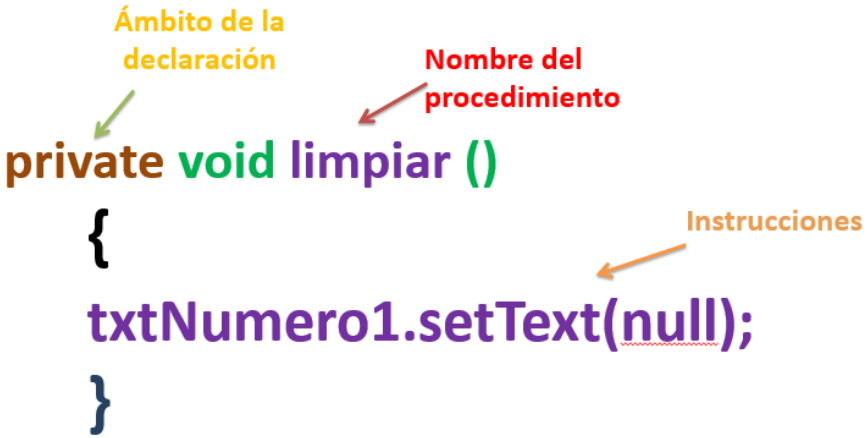


- **¿Qué es una función?** Conjunto de instrucciones agrupadas bajo un nombre y con un objetivo común, que se ejecuta al ser invocada

FUNCIONES

- Estas “funciones” se pueden definir antes o después del main. En POO se verá con otro nombre, **métodos**.
- En JAVA tenemos **funciones** y **procedimientos** según el formato de salida

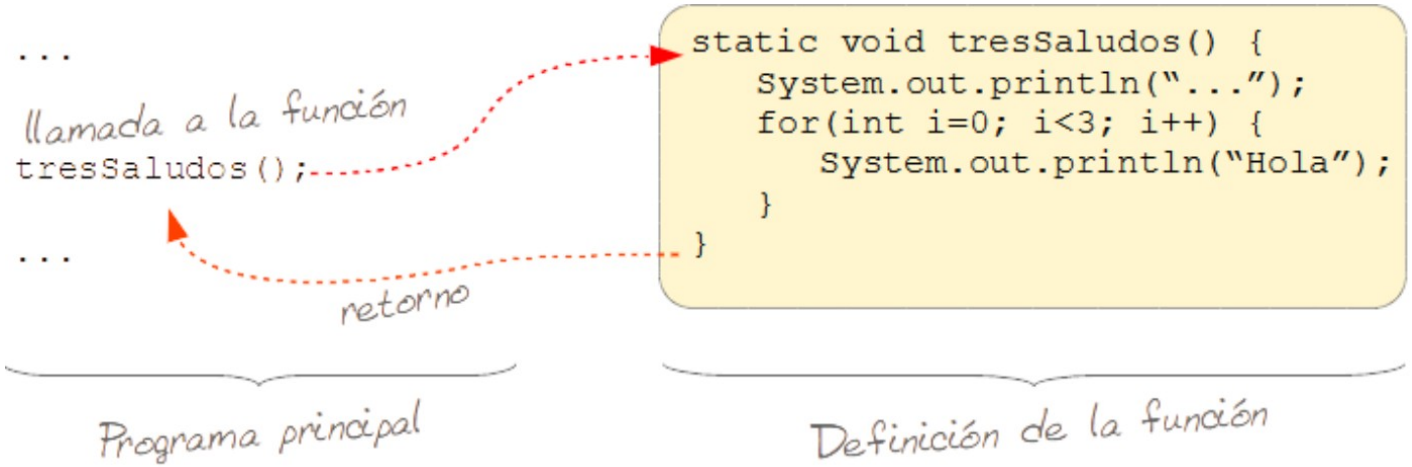




Para la definición de las “funciones” hay que indicar a parte de su nombre y parámetros: su visibilidad (public, private, protected) y si son static o no.
Los conceptos de visibilidad los entenderemos cuando veamos las clases en POO, al igual que static que determina si esa función es de la clase o del objeto. Por ahora usaremos **public** y **static**

FUNCIONES

- **¿Cómo sería el flujo de ejecución del programa?**



- **¿Puede haber dos "funciones" con el mismo nombre?** NO si se llaman exactamente igual y con el mismo número de parámetros (esto se conoce como sobrecarga de funciones)
- **Actividad:** Calcular el máximo de 3 números si existe la función maximo(a,b). Después aplica ambas para el caso de máximo de 5 números. Ten en cuenta que una función se puede utilizar en una expresión Ej: int a= 5+maximo(3,7); o int b=maximo(maximo(2,3),4);
- **Importante:** Haz las funciones por criterios de repetición o usabilidad en proyectos. No hagas funciones por hacer como tresSaludos, cuatroSaludos,... o funciones con pocas instrucciones.

FUNCIONES

- **Ámbito de una variable**

```
void func1() {
    int a, b;
    ...
    while(...) {
        int c;
        ...
    } //del while
} //de func1
```

ámbito func1. Podemos utilizar las variables: a y b

ámbito while. Podemos utilizar las variables: a, b y c

- **¿Puedo usar las variables externas a una función?** Si queremos usar una variable fuera de ámbito, como las constantes, tenemos que definirlas fuera del main o el bloque donde se vayan a usar.

FUNCIONES

- **Los parámetros en las funciones:** En la definición de las funciones se tienen que indicar los parámetros y el tipo asociado. Al invocar la función se tienen que asignar los parámetros en el mismo orden y formato adecuados.

```
public static void main(String[] args) {
    System.out.println(Saludo(a:"hola",b:"Domingo"));
}
public static String Saludo(String a, String b){
    return a+' '+b;
}
```

```
public static void main(String[] args) {
    System.out.println(prueba(a:2,(int)3.0));
}
public static int prueba(double a, int b){
    return b;
}
```

- **Actividad:** Crea un procedimiento para calcular el área y volumen de un cilindro (2*PI*(altura+radio), PI*radio^2*altura)
- **Actividad:** Crear las funciones esPar, esDivisible3 y un procedimiento para imprimir los números pares divisibles por 3 que hay entre 1 y 20

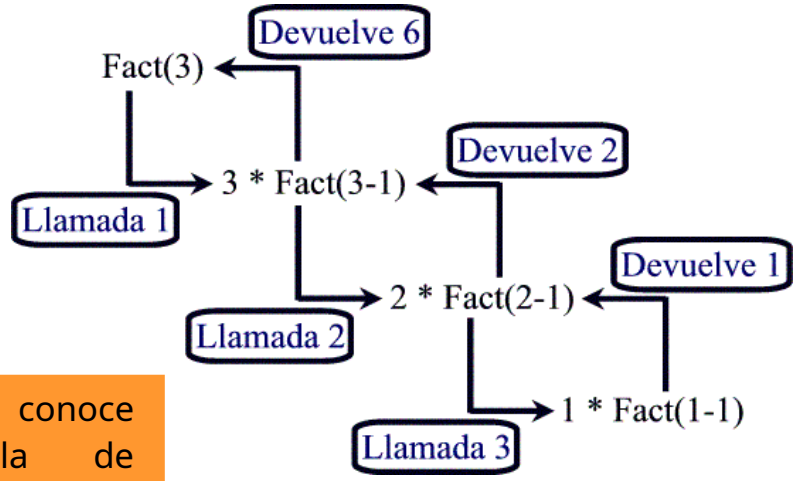
FUNCIONES

- **Recursividad:** Es el caso de que una función se llama a sí misma durante la ejecución
- La estructura de una función recursiva es la siguiente:

```
Int funcionRecursiva(datos) {
      int resultado;
      if (caso base) {
        resultado=valorBase;
      else{ //Caso recursivo
        resultado=funcionRecursiva(nuevoDato);
      }
    }
```

- Ejemplo:

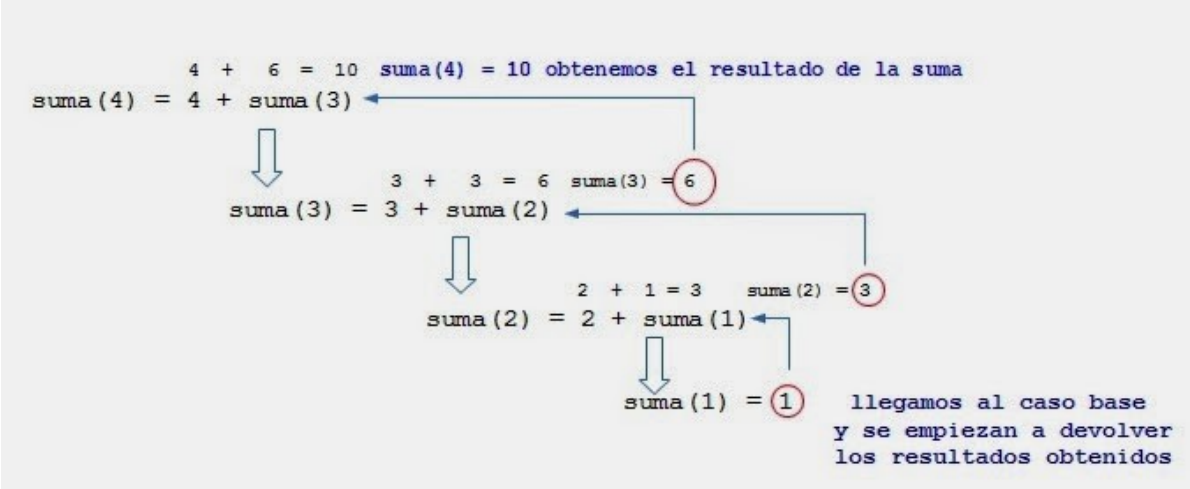
```
public static long calcularFactorial(int n) {
    // Caso base: el factorial de 0 es 1
    if (n == 0) {
        return 1;
    } else {
        // Paso recursivo: n * factorial(n-1)
        return n * calcularFactorial(n - 1);
    }
}
```



Esto se conoce como pila de llamadas recursivas

FUNCIONES

- **Recursividad:** Otro ejemplo de pila de recursividad:



- **Práctica:** Vamos a realizar la pila de llamadas de los códigos siguientes y a escribir la función recursiva del ejemplo suma(N) en JAVA

```
public static void main(String[] args) {
    int a = 254;
    System.out.println(funcion(a, 2));
}
public static int funcion(int num, int pos){
    if(num < 10){
        return num;
    }else{
        return (num % 10) * (int) Math.pow(10, pos) + (funcion(num/10, pos-1));
    }
}
```

```
public static void incre(int n){
    if(n>0){
        incre(n-1);
        System.out.print(n);
    }
    else
    System.out.println();
}
```

Cambia el orden a ver el resultado

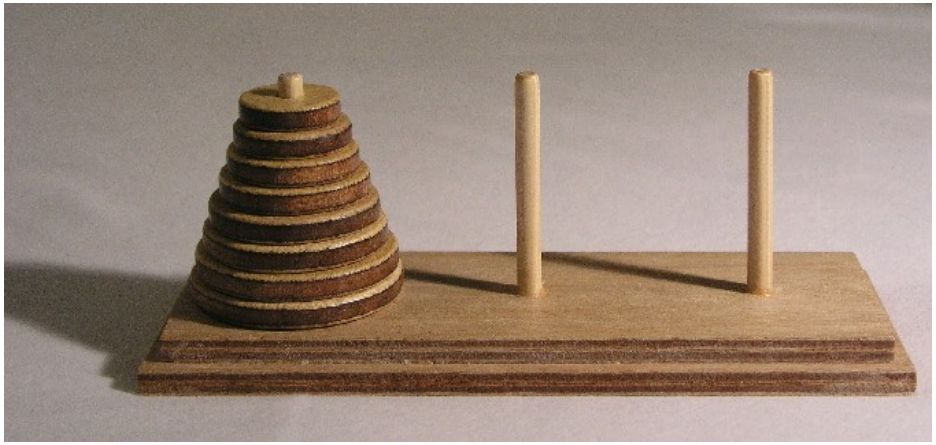
FUNCIONES

- **Actividades recursividad**:
  - Función que devuelve la cantidad de dígitos de un número positivo
  - Función que devuelve la suma de los números naturales desde 1 hasta N. Compara el código con el que se obtendría iterativo.
  - Función que calcula a^b usando recursividad. Recordad que $a^b=a*a^{b-1}$
  - Método recursivo para pasar un número decimal, que compruebe que es positivo, y pasarlo a binario mediante sucesivas divisiones por 2

Los ejemplos que se han pedido son sencillos y la recursividad es una forma de solución divide y vencerás. Eso no quita que un problema se resuelva de varias maneras

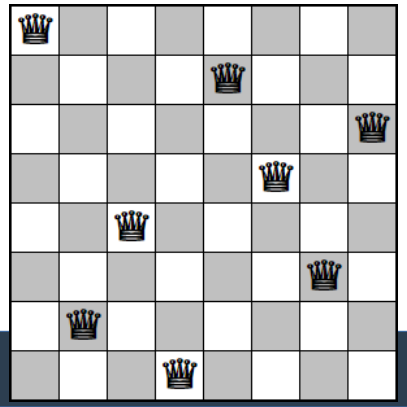
FUNCIONES

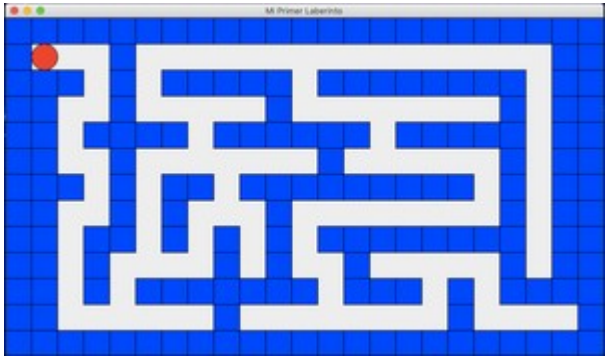
- Ejemplo que ya no podría hacerse iterativo: Torres de HANOI



```
//Método Torres de Hanoi Recursivo
    public static void Hanoi(int n, int origen,  int auxiliar, int destino){
        if(n==1){
           System.out.println("mover disco de " + origen + " a " + destino);
        }else{
            //Se llevan los n-1 de origen al auxiliar
           Hanoi(n-1, origen, destino, auxiliar);
            //Se llevan el que queda a destino
           System.out.println("mover disco de "+ origen + " a " + destino);
            //Se llevan los n-1 de auxiliar a destino
           Hanoi(n-1, auxiliar, origen, destino);
        }
    }
```

Otros ejemplos que veremos más adelante: El problema de las 8 reinas o salir de un laberinto





Tarea: Realiza las actividades propuestas en Moodle