2020-2021

ASIX-DAM

MONLAU centro de estudios

# M3-Programació 02/10/2020

**M3-** Programació

**Curs 2020-2021**


**Professor:**

***Gerard Amirian:*** ***amiriamg@monlau.es***

# **M3-** Programació

**Curs 2020-2021**

**Professors:** ***Paco Martínez, Gerard Amiriam***

**M3 inclou 6 UF's:**

**ASIX1-DAM1:**
**UF1: programació estructurada**
**UF2: disseny modular**
**UF3: fonaments de gestió de fitxers**

***DAM2:***
***UF4: programació orientada a objectes (POO)***
***UF5: POO. Llibreries de classes fonamentals.***
***UF6: POO. Introducció a la persistència en BD***

# **M3-** Programació

**Curs 2020-2021**

**Dinàmica de classes:**

- Explicació de la teoria (amb suport PPT)
- Petits exemples pràctics.
- Entrega de programes curts.
- Elaboració d'una pràctica de caràcter més global.

**Moodle:**

- Presentacions (PPT) de classe.
- PDF's d'informació addicional.
- Curs On-Line Java
- Curs On-Line Java Script

**Llibres:**

- Llibre de suport per Certificació JAVA (pdf)
- Llibre de suport (paper)

**M3-** Programació

**Curs 2020-2021**

# Fundamentos de programación en JAVA

# UF1-M3

PROGRAMACIÓN JAVA

## Programación estructurada

# M3-

```
// package: contiene un conjunto de clases que forman nuestra aplicación
package actividad1;
// import:  para importar librerías
import java.util.Scanner;
// public: nivel de visibilidad
public class ejercicio1
{ // inicio de clase
// main: clase  inicial
  public static void main(String[ ] args) {
      //Código a ejecutar
  } //fin de main
}//fin de clase
```

# M3-

-CODIGO FUENTE:

El código de un programa escrito en un lenguaje de programación.

El compilador se encarga de transformarlo en un código ejecutable por una máquina concreta.

Java: Es un lenguaje de programación genera código bytecode, fichero con la extensión .class para una maquina virtual

# M3-

**Comentarios multi-línea:**

**Comienzan con /* y terminan con */**

**Comentarios de una sola línea : //**
**Ejemplo:**

```
//Esto es un comentario
/* Esto también es
      un  comentario */
```

# (help) Java : escribir en pantalla

## Para escribir en la pantalla:

```
System.out.print(" texto" );
//escribe el texto en la pantalla
```

```
System.out.println(" texto" );
/*escribe el texto en la pantalla y situa el cursor en
la siguiente línia*/
```

```
System.out.println(" texto1"  + " texto2"  );
```

*+ : es concatenar o añadir*

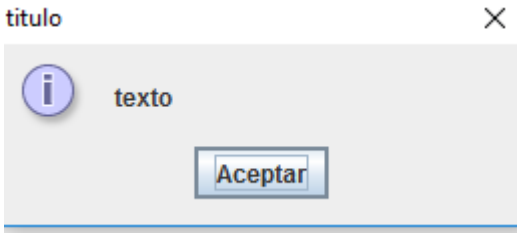
# (help) Java : el teclado

**Incorporar Librería:** `import java.util.Scanner;`

```
/en la zona de declarciones globales definimos una instancia del teclado con el
nombre por ejemplo teclado:
static Scanner keyboard=new Scanner(System.in);
// es conveniente indicar que un dato termina con newLine:
 keyboard.useDelimiter("\n" );

// y para leer String:
String text=keyboard.next();
// para leer int:
int dataInt=keyboard.nextInt();

// para Leer double
double dataDouble=keyboard.nextDouble();

// para Leer float
float dataFloat=keyboard.nextFloat();

// para Leer boolean
boolean dataBoolean=keyboard.nextBoolean();

// para Leer char
char dataChar=(keyboard.next()).charAt(0);
```

# (help) Java : escribir en pantalla

**JOptionPane.showMessageDialog ( null," texto" ," titulo" , JOptionPane.INFORMATION_MESSAGE );**




**Ejemplo:**
**JOptionPane.showMessageDialog ( null, " ¡Bienvenido a Java!" , " Programa de ejemplo" , JOptionPane.INFORMATION_MESSAGE );**

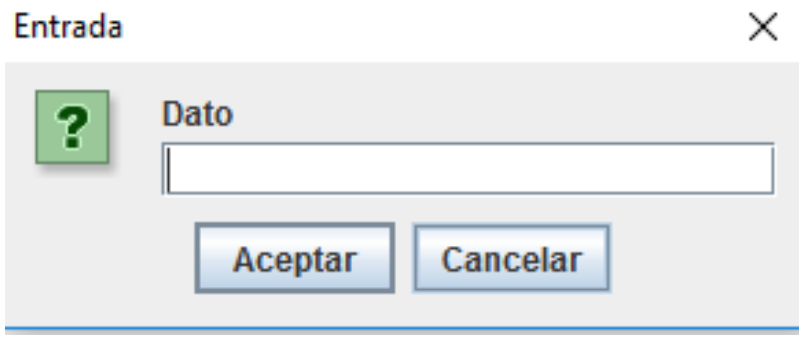
# (help) Java :Leer datos

**JOptionPane.showInputDialog(" Dato" );** **devuelve un dato String**

**Para leer un dato de tipo string:**
**String DatoString=JOptionPane.showInputDialog(" Dato" );**

**Para leer un dato de tipo int:**
**int DatoInt=Integer.parseInt(JOptionPane.showInputDialog(" Dato" ));**

**Para leer un dato de tipo double:**
**double datoDouble=Double.parseDouble(JOptionPane.showInputDialog(" Dato" ));**



# - Operaciones matemáticas

| Operador | Nombre | Descripción |
|---|---|---|
| + | Suma | 5+2 → 7 |
| - | Resta | 5-2 → 3 |
| * | Multiplicación | 5*2 → 10 |
| / | División | 5/2 → 2 |
| % | Residuo | 7%2 → 1 |

# -Variables (1)

| | |
|---|---|
| **int** | **Números enteros** |
| **long** | **Números enteros grande** |
| **float** | **Números reales** |
| **double** | **Números reales: doble precisión** |
| **char** | **Carácter** |
| **String** | **Cadena de Caracteres** |
| **boolean** | **boleana (false true)** |

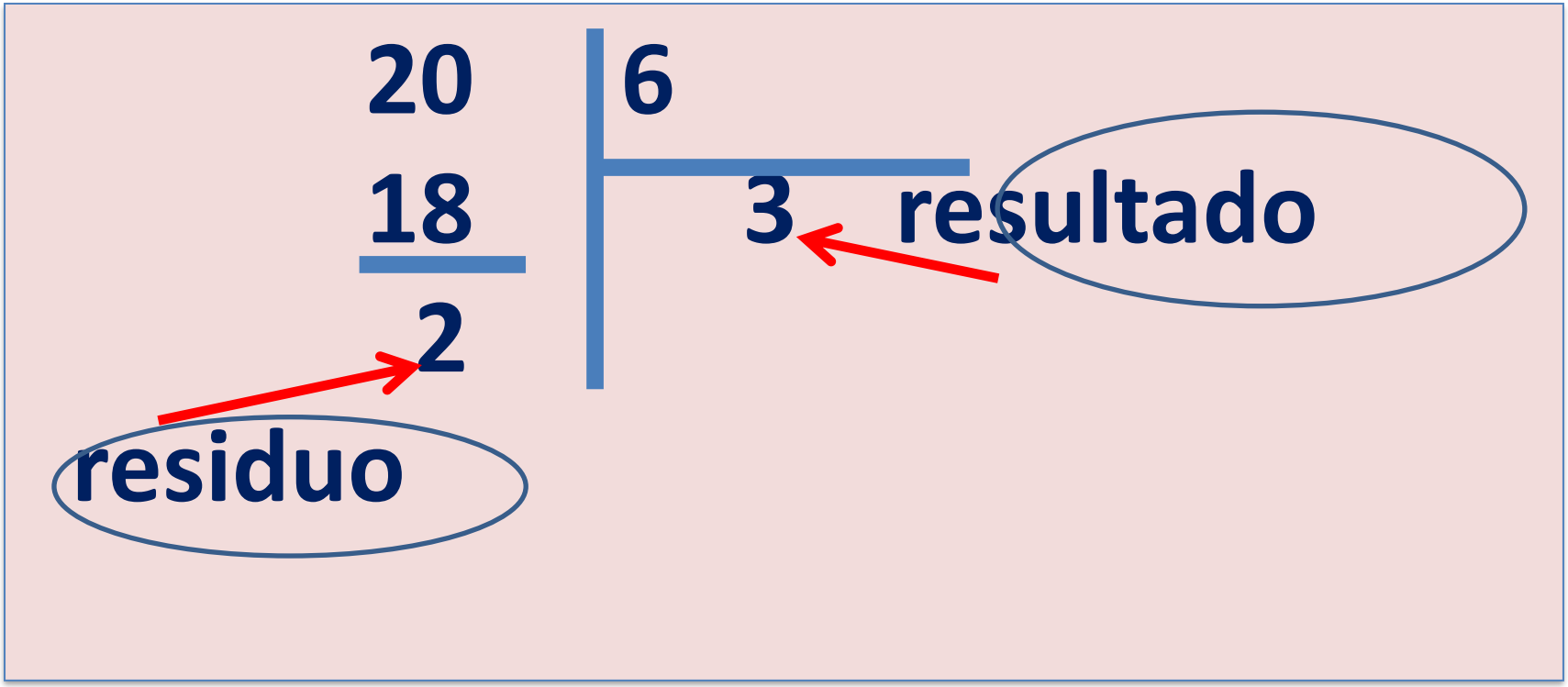
# Operaciones matemáticas

| Operador | Nombre | Descripción |
|---|---|---|
| + | Suma | 5+2 → 7 |
| - | Resta | 5-2 → 3 |
| * | Multiplicación | 5*2 → 10 |
| / | División | 5/2 → 2 |
| % | Resta de una división | 20%6 → 2 |

# Operaciones incremental - decremental

| Operador | Descripción | ejemplo |
|---|---|---|
| ++ | Incremento en 1 | i++→ i=i+1 |
| -- | Decremento en 1 | i-- → i= i-1 |

# El operador residuo%




**X % Y da residuo de la división X entre Y**

**R= 20 % 3 ; // da el residuo: 2**

**R= 20 % 7 // da el resultado: 6**

# El operador %

```
int   res, x=20, y=3, d;
d    = 20/3;   //es el  resultado da   6
res= 20 % 3; // es el residuo  da    2
```

# Operaciones Lógicas o relacionales (comparaciones )

| Operador | Nombre | Descripción |
|---|---|---|
| == | Igual a | if (a==‘s’) |
| != | Diferente de | if (a!=null) |
| > | Mayor que | if (a>0.5) |
| < | Menor que | if (a<2l) |
| >= | Mayor o igual que | if (a>=2) |
| <= | Menor o igual que | if (a<=3) |

# Operaciones Lógicas  o relacionales (comparaciones  )

en el caso de comparar 2 strings:

***var1.equals(var2)*** ***en lugar de:***

***var1 ==var2***

# -Estructuras de control
# if  else

```
if    (condición )
{
   sentencias; // se ejecutarán si se cumple la condición
}
else
{
   sentencias; // se ejecutarán SI NO se cumple la condición
}
```

# -Estructuras de control
# if  else

**Caso particular: if sin else**

```
if    (condición )
{
   sentencias; //  se ejecutarán si se cumple la condición
}
```

*… y sino  sigue la siguiente instrucción*

# -Estructuras de control
# if  else

**Caso particular:**
**Si solo se debe ejecutar una sentencia  no hace falta poner {  }**

```
if   (condicion ) { sentencia; }
```

**Es lo mismo que:**

```
if   (condicion )  sentencia;
```

# -Estructura: if

**Lista de operadores lógicos aplicable a if :**

**==** → significa **si son iguales**

equals en el caso de comparar 2 strings

**&&** → significa **y** (and): 2 o más condiciones que deben cumplir todas

**||** → significa **O** (or): 2 o más condiciones que basta cumplir una

**!** → significa negación **no** (not)

**!=** → significa **NO son iguales: diferentes**

# -Estructuras de control
# if  else

**if(condición1 && condición2)** :

**(deben cumplirse todas las condiciones )**

**O sea si se cumple la condición1**

**Y ADEMÁS se cumple la condición2**

**entonces….**

# -Estructuras de control
# if else

**if(condición1 || condición2) :**

**(deben cumplirse al menos una de las condiciones )**

**O sea si se cumple alguna de las condiciones,**

**condición1 O condición2 o las 2, entonces….**

# -Estructuras de control
# if  else

**if( !condición1){  }**

**significa si: NO se cumple  la condición1**

**entonces….**

# - Expresiones booleanas

## Operadores booleanos:

Se deben cumplir las condiciones:

**A&&B: AND**

Basta que se cumpla una condición:

**A||B: OR**

**! Inversa de una condición**

**!A : NOT**

# Combinaciones de IF-ELSE

```
if (condicion1)  {
        //tarea que se realizara si se cumple condicion1
         }
 else if (condicion2)     {
            /*tarea que se realizara si No se cumple condicion1
              * y se cumple condicion2 */
        }
```

# Combinaciones de IF-ELSE

```
if (condicion1)  {
        if (condicion2)     {
      /*tarea que se realizara si se cumple condicion1
            * y condicion2 */
        }
    }
```

---

La combinación anterior es equivalente a:

```
if (condicion1  && condicion2)     {
          /*tarea que se realizara si No se cumple condicion1
            * y se cumple condicion2 */
        }
```

# Combinaciones de IF-ELSE

```
if (condicion1 == true)  {
        //tarea que se realizara si se cumple condicion1
   }
```

---

La estructura condicional  anterior es equivalente a:

```
if (condicion1)  {
        //tarea que se realizara si se cumple condicion1
   }
```

# Combinaciones de IF-ELSE

```
if (condicion1 == false)  {
        //tarea que se realizara si NO se cumple condicion1
   }
```

---

La estructura condicional  anterior es equivalente a:

```
if (!condicion1)  {
        //tarea que se realizara si se NO  cumple condicion1
   }
```

# ejemplos de IF-ELSE

```
if (edad>=18 && edad <=100)  {
      System.out.print(" puedes conducir" );
}
if (edad<18 || edad >100)  {
      System.out.print(" NO puedes conducir" );
}
```

---

Por lo tanto se puede observar que llegas a lo mismo si inviertes todas las lógicas ( :

(*edad >= 18  && edad <=100*)  es igual que

! (*edad <  18   ||  edad >100*)

---

Las leyes de De Morgan representan esta lógica:

**" *no (A y B)*"  es lo mismo que " *(no A) o (no B)*"**

**y también,**

**" *no (A o B)*"  es lo mismo que " *(no A) y (no B)*"**

# -Estructura switch

## - SWITCH

**Estructura para hacer un menú de usuario o elegir una opción entre varias.**

```
switch(variableDeControlDeLaopcion)
{
    case valor1: //sentencias del caso 1
                                    break; //salir del switch
    case valor2: //sentencias del caso 2
                                    break; //salir del switch


   default: //sentencias de los  casos NO   previstos
}
```

# Práctica

**Realizar un programa con las siguientes opciones:**
**1)- (Apto/No Apto): Pedir la nota del user y decir si está aprobado o no.**

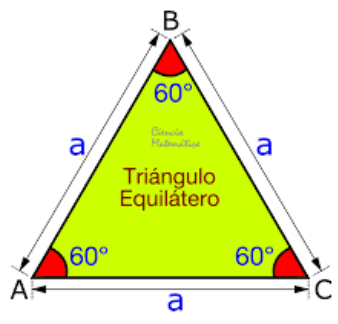

**2)- (carnet de conducir): Pedir la edad del user y decir si puede sacar el carnet de conducir o no.**

**3)-Pide un número entero y determinar:**
**-si es 7 o no //**
**-si es mayor, menor o igual a 100. //**
**-si es par o impar**

**-si es divisible entre 7**
**-si está entre 10 y 25(ambos incluidos)**


**4)- (ganador): En un partido de futbol entre 2 equipos (Local y Visitante), //Aaron pedir el número de goles marcados por cada equipo y decir qué equipo ha ganado o si han empatado.**

**5)- (triángulo): Pedir los 3 lados de un triángulo y decir si es equilátero (los 3 lados iguales)**



# *Bucles*-ITERACIÓN-Estructura while

**REPETICIONES (BULCLES)**

**- Bucle while**

***while se encarga de repetir un bloque de código mientras se cumpla la condición.***

***-Se usa while cuando no se sabe el número de repeticiones***