















NTT Data

JAVA BÁSICO

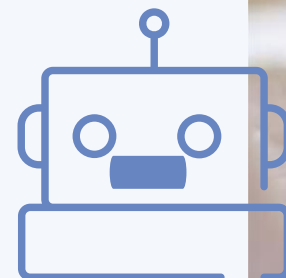
Abril 2022

**FUTURE
AT HEART**

INDICE

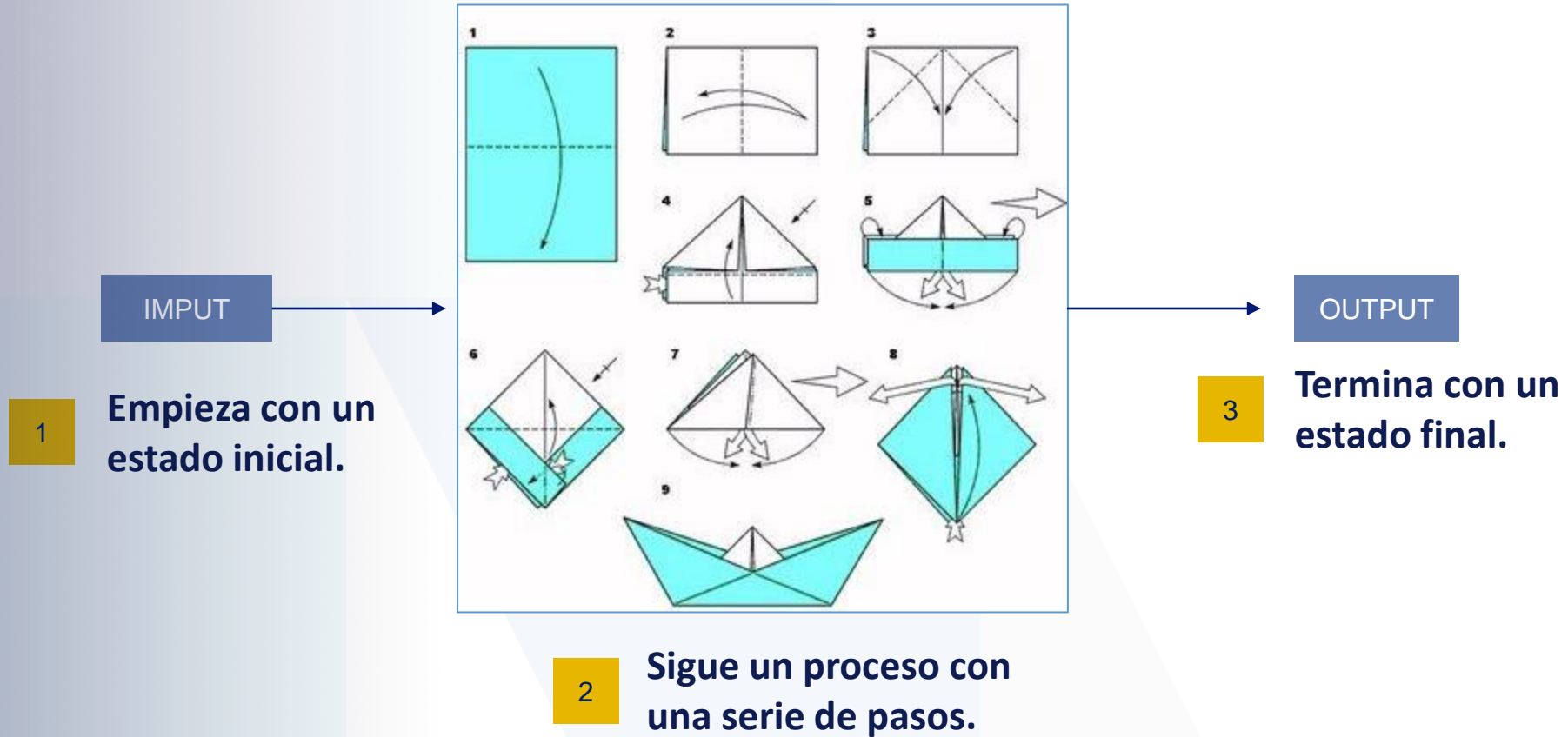
- 1  ¿QUÉ ES UN ALGORITMO?
- 2  ¿QUÉ ES LA PROGRAMACIÓN INFORMÁTICA?
- 3  ¿QUÉ ES JAVA?
- 4  CREACIÓN DE PROYECTO
- 5  VARIABLES
- 6  OPERACIONES ARITMETICAS
- 7  ESTRUCTURAS DE CONDICIÓN
- 8  ESTRUCTURAS DE ITERACIÓN
- 9  CADENAS DE CARACTERES
- 10  ARREGLOS
- 11  MATRICES
- 12  TE TOCA TI ... (EJERCICIO FINAL)

NTT Data



1. ¿QUÉ ES UN ALGORITMO?

ES UNA SERIE DE SECUENCIA DE INSTRUCCIONES

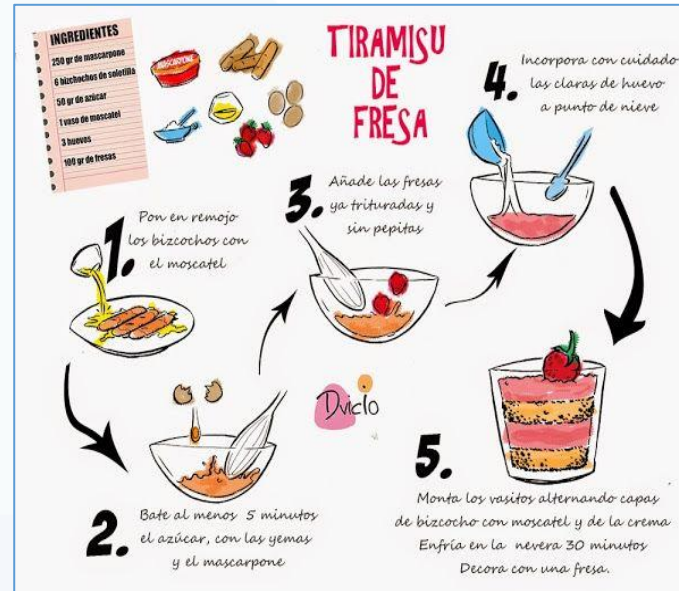


A. EJEMPLOS DE ALGORITMOS COTIDIANOS

NTT DATA



¿Cómo recargar una tarjeta del metropolitano?

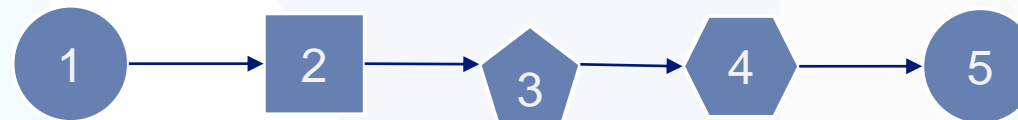


¿Cómo preparar mi plato favorito?



¿Cómo pedir un taxi por aplicación?

¿QUÉ TIENEN EN COMÚN LOS 3 CASOS?



2.¿QUÉ ES LA PROGRAMACIÓN INFORMÁTICA?

ES LA CIENCIA QUE SE ENCARGA DE INDICAR AL ORDENADOR QUE HACER, MEDIANTE UNA SERIE DE INSTRUCCIONES CONOCIDAS COMO ALGORITMOS

```
1  Proceso sin_titulo
2      totalfactura<-0;
3      Escribir "Ingrese la fecha actual";
4      Leer fecha;
5      Escribir "Ingrese el nombre del cliente";
6      Leer cliente;
7      Escribir "Ingrese la direccion del cliente";
8      Leer direccion;
9      Escribir "Ingrese la cantidad de detalles";
10     Leer detalles;
11     Para i<-1 Hasta detalles Con Paso 1 Hacer
12         Escribir "Ingrese la cantidad";
13         Leer cantidad;
14         Escribir "Ingrese la descripción";
15         Leer descripcion;
16         Escribir "Ingrese el precio";
17         Leer precio;
18         total<-cantidad*precio;
19         totalfactura<-totalfactura+total;
20     FinPara
21     Escribir totalfactura;
22 FinProceso
```

Ejemplo de pseudocódigo

```
public class repetitivo3 {
    public static void main(String[] args) {
        // TODO code application logic here
        int num, cont, pares, impares;
        pares = 0;
        impares = 0;
        Scanner ingreso=new Scanner(System.in);
        for(cont = 1; cont < 11; cont++){
            System.out.print("Ingrese número " + cont + " : ");
            num = Integer.parseInt(ingreso.next());
            if((num - ((num / 2)*2)) == 0){
                pares++;
            }else{
                impares++;
            }
        }
        System.out.println("PARES INGRESADOS : " + pares);
        System.out.println("IMPARES INGRESADOS : " + impares);
    }
}
```

Ejemplo de Código Fuente

3. ¿QUÉ ES JAVA?

JAVA ES UN TIPO DE LENGUAJE DE PROGRAMACIÓN Y UNA PLATAFORMA INFORMÁTICA, CREADA Y COMERCIALIZADA POR SUN MICROSYSTEMS EN EL AÑO 1995.

C A R A C T E R I S T I C A S

SIMPLE

Java ofrece la funcionalidad de un lenguaje potente, derivado de C y C++, pero sin las características menos usadas y más confusas de estos, haciéndolo más sencillo.

ORIENTADO A OBJETOS

El enfoque orientado a objetos (OO) es uno de los estilos de programación más populares. Permite diseñar el software de forma que los distintos tipos de datos que se usen estén unidos a sus operaciones.

MULTIHILO

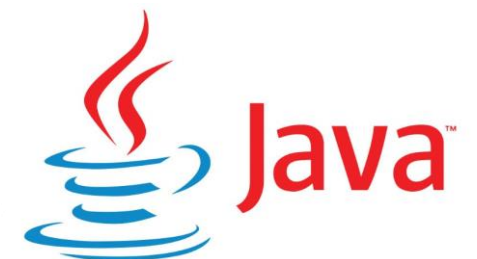
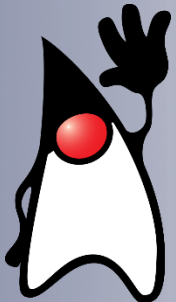
Java logra llevar a cabo varias tareas simultáneamente dentro del mismo programa. Esto permite mejorar el rendimiento y la velocidad de ejecución.

DISTRIBUIDO

Java proporciona una gran biblioteca estándar y herramientas para que los programas puedan ser distribuidos.

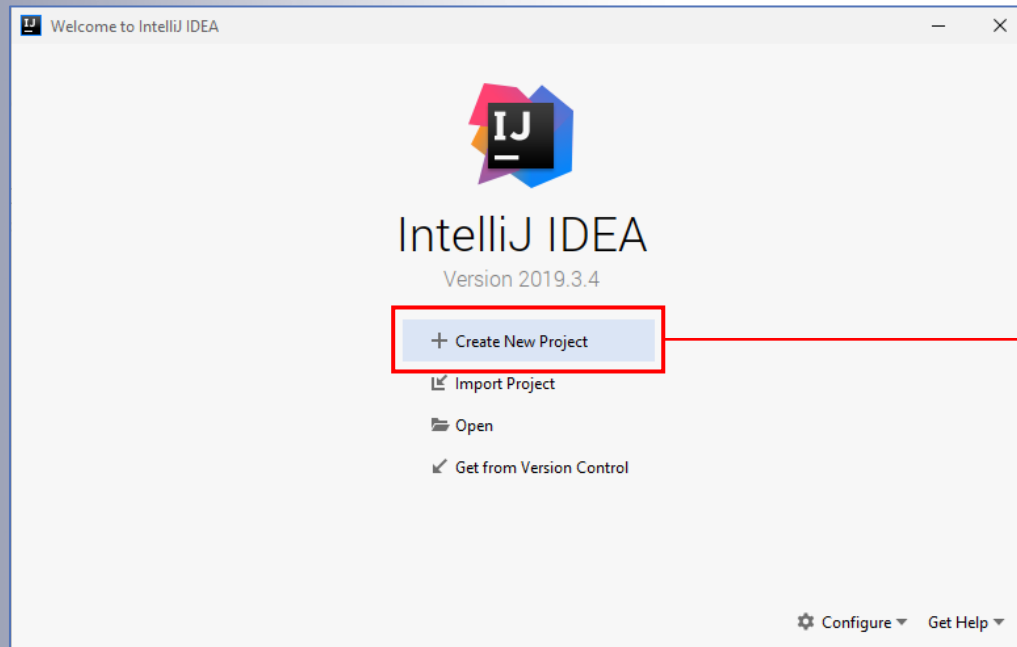
INDEPENDIENTE DE PLATAFORMA

Esto significa que programas escritos en el lenguaje Java pueden ejecutarse en cualquier tipo de hardware, lo que lo hace portable.



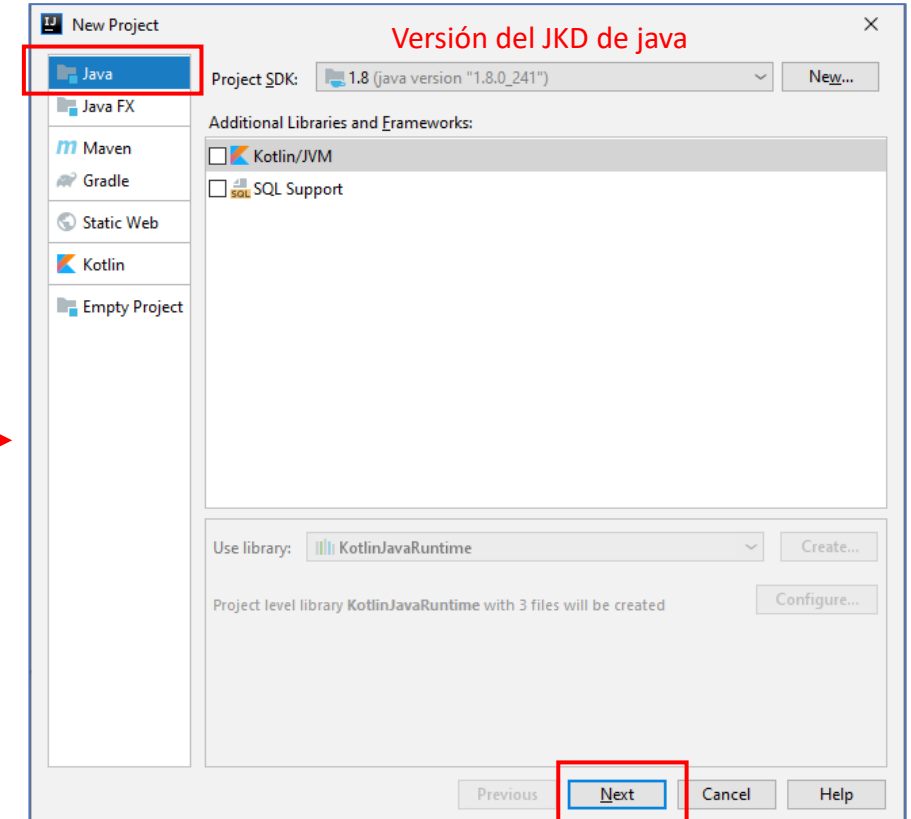
4. CREACIÓN DE PROYECTO

USAREMOS INTELLIJ IDEA COMO NUESTRO ENTORNO DE DESARROLLO INTEGRADO (IDE)



1

Seleccionamos **Crear Nuevo Proyecto**.

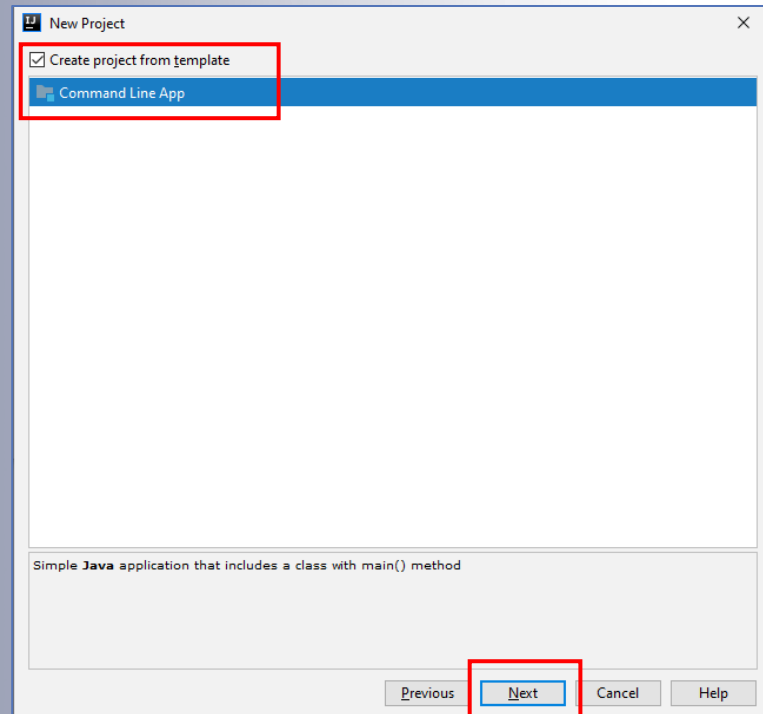


2

Seleccionamos **Siguiente**.

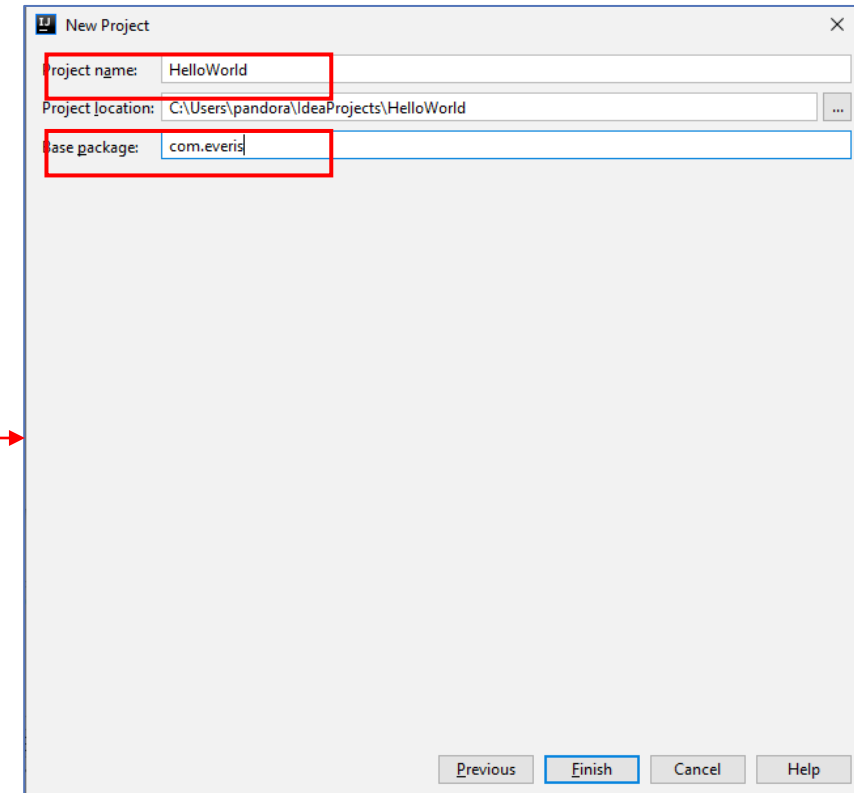
4. CREACIÓN DE PROYECTO

USAREMOS INTELLIJ IDEA COMO NUESTRO ENTORNO DE DESARROLLO INTEGRADO (IDE)



3

Seleccionamos el check de **Crear Proyecto desde plantilla** y luego **Siguiente**



4

Ingresamos el **Nombre del Proyecto**
Ingresamos el Nombre del **Paquete base**

4. CREACIÓN DE PROYECTO

The screenshot shows the IntelliJ IDEA IDE interface. The Project tool window on the left displays the project structure: 'HelloWorld' (C:\Users\pandora\Idea\HelloWorld) contains a 'src' directory, which contains a 'com.everis' package, which contains a 'Main' class. The 'Main' class is selected. The code editor shows the following Java code:

```
1 package com.everis;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         // write your code here
7     }
8 }
9
```

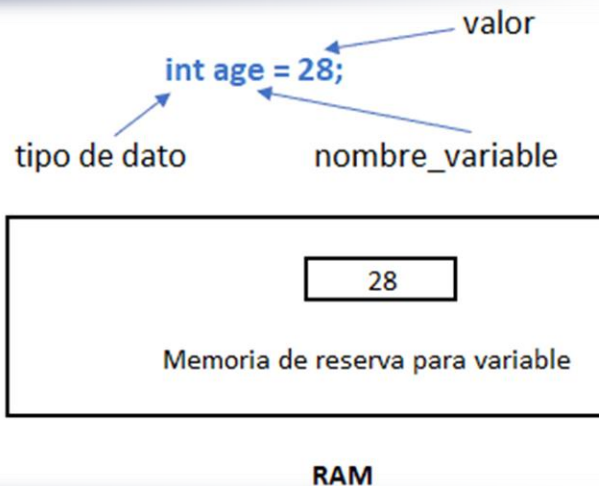
Annotations with red arrows point to the following elements:

- Nombre del Proyecto**: Points to the 'HelloWorld' project name in the Project tool window.
- Nombre del Paquete**: Points to the 'com.everis' package name in the Project tool window.
- Clase para escribir código JAVA**: Points to the 'Main' class in the Project tool window.

5. VARIABLES

¿QUÉ ES?

UNA VARIABLE ES EL NOMBRE DADO A UNA UBICACIÓN DE MEMORIA. ES LA UNIDAD BÁSICA DE ALMACENAMIENTO EN UN PROGRAMA. EL VALOR ALMACENADO EN UNA VARIABLE SE PUEDE CAMBIAR DURANTE LA EJECUCIÓN DEL PROGRAMA.



NTT Data

Es **importante** que a partir de este punto, repitas los ejemplos que se mencionarán en tu maquina.

5. VARIABLES

1 TIPOS DE VARIABLES

```
public static void main(String[] args) {  
  
    // DECLARACIÓN DE VARIABLES  
    int numero;    //Representa todos los numeros enteros  
    double decimal; //Representa todos los numeros decimales  
    boolean flag;  //Representa un valor verdadero o false (true, false)  
    String cadena; //Representa una cadena de caracteres
```

2 INICIALIZAR VARIABLES

```
// INICIALIZAR VARIABLES  
numero = 50;  
decimal = 20.0;  
flag = true;  
cadena = "Curso de java basico";
```

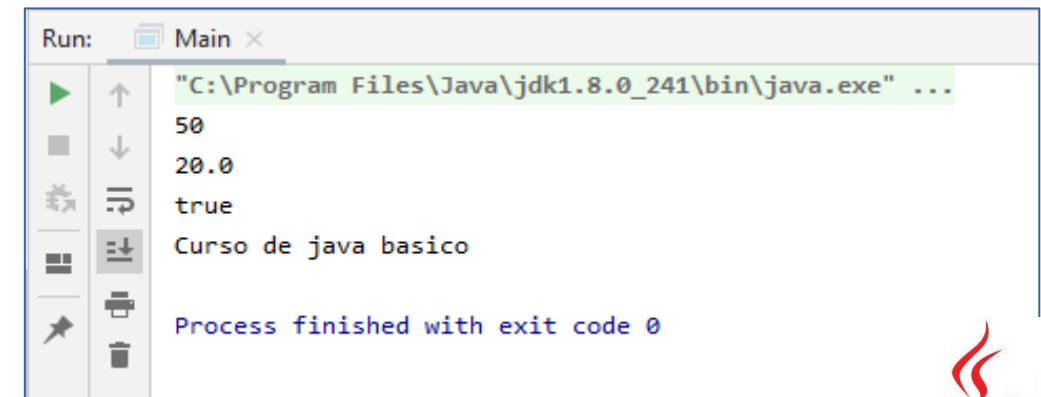
Se asigna un valor con el símbolo = a la variable según su tipo permitido.

3 EJECUCIÓN

Se utilizará el método de impresión de java para mostrar el resultado de las variables previamente inicializadas.

```
// EJECUCIÓN  
System.out.println(numero);  
System.out.println(decimal);  
System.out.println(flag);  
System.out.println(cadena);
```

Revisaremos en la terminal los resultados de la ejecución



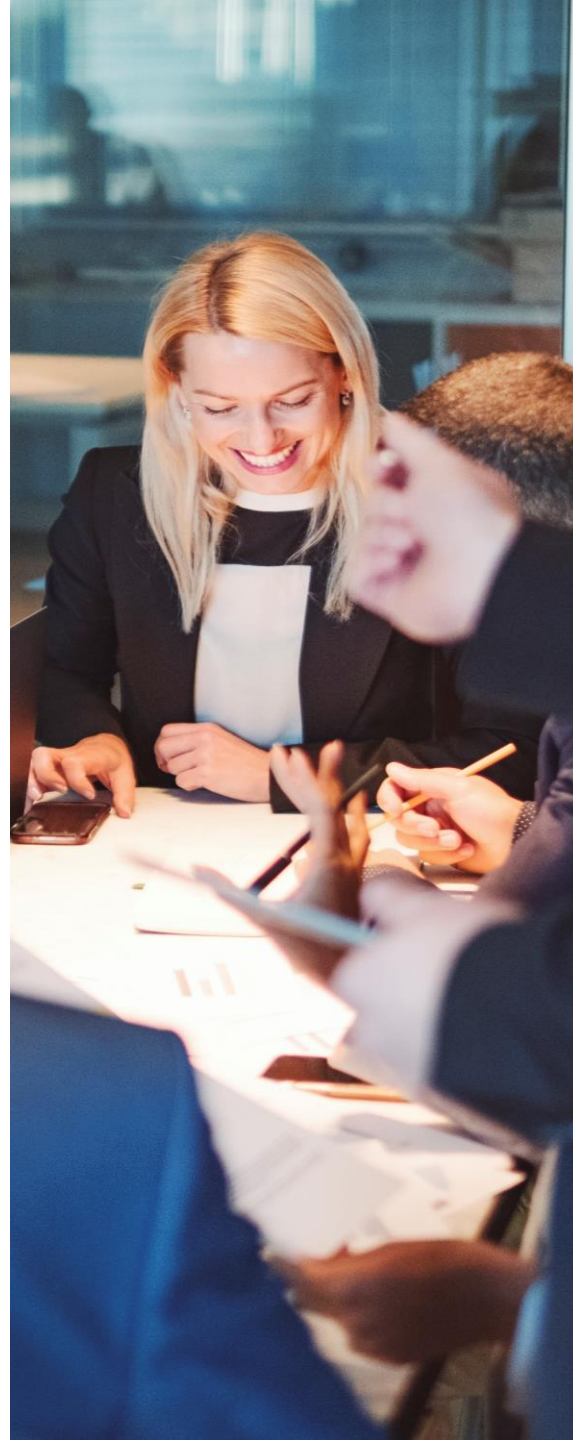
```
Run: Main x  
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...  
50  
20.0  
true  
Curso de java basico  
  
Process finished with exit code 0
```



6. OPERACIONES ARITMETICAS

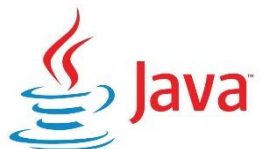
¿QUÉ ES?

LOS OPERADORES ARITMÉTICOS EN JAVA SON LOS OPERADORES QUE NO PERMITE REALIZAR OPERACIONES MATEMÁTICAS: SUMA, RESTA, MULTIPLICACIÓN, DIVISIÓN Y RESTO. LOS OPERADORES ARITMÉTICOS EN JAVA LOS UTILIZAREMOS ENTRE DOS LITERALES O VARIABLES Y EL RESULTADO, NORMALMENTE LO ASIGNAREMOS A UNA VARIABLE O BIEN LO EVALUAMOS.



NTT DATA

- ✓ **Suma:** usar el operador “+”
- ✓ **Resta:** usar el operador “-”
- ✓ **Multipliación:** usar el operador “*”
- ✓ **División:** usar el operador “/”
- ✓ **Resto :** usar el operador “%”
Es el resto de una división entre enteros
(en otros lenguajes denominado mod)



6. OPERACIONES ARITMETICAS

SUMA

```
//DECLARACION
int numeroA;
int numeroB;

//INCIALIZACIÓN
numeroA =5;
numeroB =8;

System.out.println(numeroA+numeroB);
```

```
Run: Main x
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe"
13
Process finished with exit code 0
```

EL OPERADOR ES: " + "

RESTA

```
//DECLARACION
int numeroA;
int numeroB;

//INCIALIZACIÓN
numeroA =8;
numeroB =3;

System.out.println(numeroA-numeroB);
}
```

```
Run: Main x
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe"
5
Process finished with exit code 0
```

EL OPERADOR ES: " - "

MULTIPLICACIÓN

```
//DECLARACION
int numeroA;
int numeroB;

//INCIALIZACIÓN
numeroA =2;
numeroB =5;

System.out.println(numeroA*numeroB);
}
```

```
Run: Main x
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe"
10
Process finished with exit code 0
```

EL OPERADOR ES: " * "



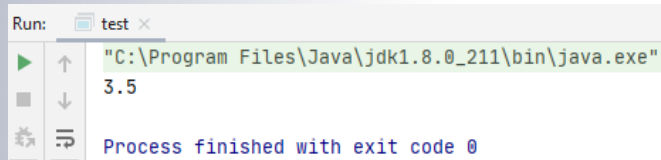
6. OPERACIONES ARITMETICAS

DIVISIÓN

```
//DECLARACIÓN
double numeroA;
double numeroB;

//INICIALIZACIÓN
numeroA =7;
numeroB =2;

System.out.println(numeroA/numeroB);
```



EL OPERADOR ES: “ / ”

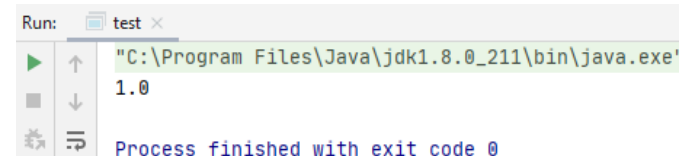
¡En este caso si declara “Int” el resultado mostrado sería “3”. **Cuidado con los tipos de variables usados.!**

RESTO

```
//DECLARACIÓN
double numeroA;
double numeroB;

//INICIALIZACIÓN
numeroA =7;
numeroB =2;

System.out.println(numeroA%numeroB);
```



EL OPERADOR ES: “ % ”

$7 \div 2 = 3.5$ (e.g división entera : 3 + resto 1)
 $7 \% 2$ Retorna el resto de la división entera de 7 por 2 (es decir 1 dado)

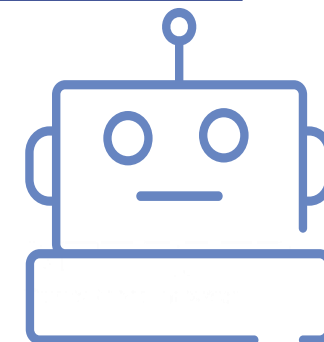
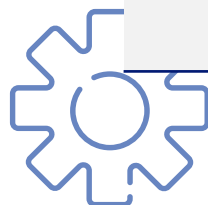


7. ESTRUCTURAS DE CONDICIÓN

¿QUÉ ES?

LOS OPERADORES LÓGICOS NOS PROPORCIONAN UN RESULTADO A PARTIR DE QUE SE CUMPLA O NO UNA CIERTA CONDICIÓN, PRODUCEN UN RESULTADO BOOLEANO, Y SUS OPERANDOS SON TAMBIÉN VALORES LÓGICOS O ASIMILABLES A ELLOS (LOS VALORES NUMÉRICOS SON ASIMILADOS A CIERTO O FALSO SEGÚN SU VALOR SEA CERO O DISTINTO DE CERO).

OPERADOR	DESCRIPCIÓN
==	Es igual
!=	Es distinto
<, <=, >, >=	Menor, menor o igual, mayor, mayor o igual
&&	Operador and (y)
	Operador or (o)
!	Operador not (no)



7. ESTRUCTURAS DE CONDICIÓN

OPERADORES LÓGICOS

//DECLARACIÓN

```
int numeroA;
```

```
int numeroB;
```

//INCIALIZACIÓN

```
numeroA = 2;
```

```
numeroB = 5;
```

//OPERADORES LÓGICOS

```
System.out.println("Número A es igual a Número B");
```

```
System.out.println(numeroA == numeroB);
```

```
System.out.println("Número A es distinto a Número B");
```

```
System.out.println(numeroA != numeroB);
```

```
System.out.println("Número A es menor a Número B");
```

```
System.out.println(numeroA < numeroB);
```

```
System.out.println("Número A es mayor a Número B");
```

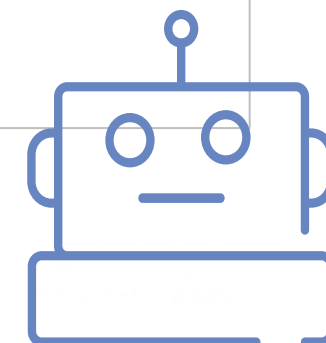
```
System.out.println(numeroA > numeroB);
```

```
System.out.println("Número A no es igual a Número B");
```

```
System.out.println(numeroA != numeroB);
```

LA EJECUCIÓN NOS DICE: SI EL RESULTADO DE LAS OPERACIONES SON VERDADERAS O FALSAS

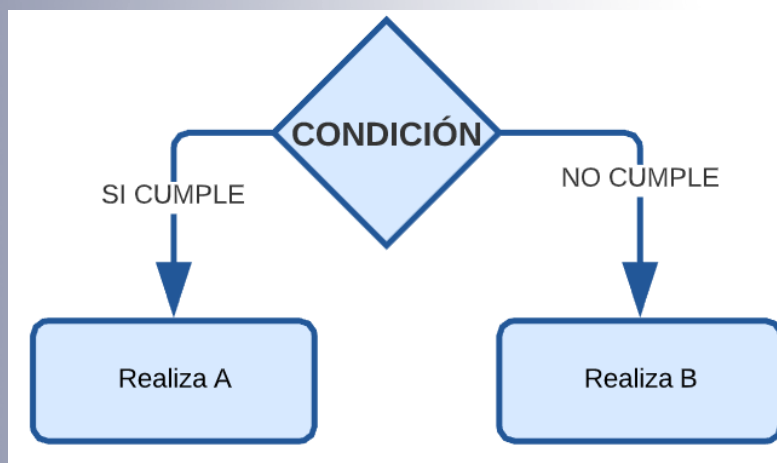
```
Run: Main x
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...
Número A es igual a Número B
false
Número A es distinto a Número B
true
Número A es menor a Número B
true
Número A es mayor a Número B
false
Número A no es igual a Número B
true
Process finished with exit code 0
```



7. ESTRUCTURAS DE CONDICIÓN

SENTENCIA IF

LA SENTENCIA **IF**, ES UNA ESTRUCTURA DE CONTROL, QUE NOS PERMITEN TOMAR CIERTA DECISIÓN AL INTERIOR DE NUESTRO ALGORITMO, ES DECIR, NOS PERMITE DETERMINAR QUE ACCIONES TOMAR CUANDO UNA CONDICIÓN SI CUMPLE.



```
//DECLARACIÓN  
int numeroA;  
int numeroB;
```

```
//INICIALIZACIÓN  
numeroA = 10;  
numeroB = 5;
```

```
//SENTENCIA IF
```

```
if (numeroA == numeroB) { // Es igual a  
    System.out.println("Realizo SUMA");  
    System.out.println(numeroA + numeroB);  
}  
if (numeroA != numeroB) { // No es igual a  
    System.out.println("Realizo RESTA");  
    System.out.println(numeroA - numeroB);  
}
```

Resultado es falso

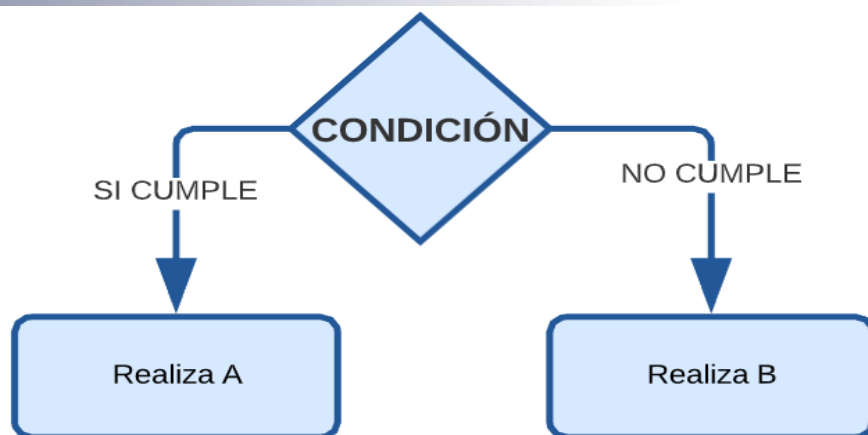
Resultado es verdadero

The screenshot shows a Java IDE's run window. The command prompt shows the execution of the Java program, which outputs "Realizo RESTA" and "5". The process finished with exit code 0.

7. ESTRUCTURAS DE CONDICIÓN

SENTENCIA ELSE

LAS SENTENCIAS **ELSE**, ES UNA ESTRUCTURA DE CONTROL, QUE NOS PERMITEN TOMAR CIERTA DECISIÓN AL INTERIOR DE NUESTRO ALGORITMO, ES DECIR, NOS PERMITE DETERMINAR QUE ACCIONES TOMAR CUANDO UNA CONDICIÓN NO CUMPLE. TENER EN CUENTA QUE ESTE SENTENCIA TIENE QUE IR INMEDIATAMENTE DESPUES DE LA SENTENCIA IF.



```
//DECLARACIÓN
int numeroA;
int numeroB;

//INCIALIZACIÓN
numeroA = 10;
numeroB = 5;

//SENTENCIA IF

if (numeroA == numeroB) { // SI ES VERDADERO, HACE SUMA
    System.out.println("Realizo SUMA");
    System.out.println(numeroA + numeroB);
} else { // SINO, HACE RESTA
    System.out.println("Realizo RESTA");
    System.out.println(numeroA - numeroB);
}
```

El resultado es el mismo que el ejemplo anterior, hemos mejorado el código con la sentencia **ELSE**

```
Run: Main x
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...
Realizo RESTA
5
Process finished with exit code 0
```

7. ESTRUCTURAS DE CONDICIÓN

SENTENCIA ANIDADAS

LA SENTENCIAS ANIDADAS SON EL CONJUNTO DE LAS SENTENCIAS PREVIAMENTE REVISAS **IF Y ELSE**, AHORA REVISAREMOS **ELSE IF** EL NOS PERMITE DETERMINAR QUE ACCIONES TOMAR CUANDO UNA CONDICIÓN PREVIA **NO CUMPLE**.

En el siguiente ejemplo demos leerlo de la siguiente manera:

1. Si el tiempo **es igual** a noche, entonces pregunta:

- A - **Si** la temperatura es mayor o igual a 26, entonces es un día caluroso
- B - **SINO**, la temperatura debe ser mayor o igual a 22 y la temperatura debe ser menor o igual a 25, entonces es un día cálido.
- C - **SINO**, la temperatura debe ser mayor o igual a 17 y la temperatura debe ser menor o igual a 21, entonces es un día frío.
- D - **Si** ninguna condición cumple, entonces la temperatura no coincide.

2. Si el tiempo no es noche, entonces es de día.

```
// DECLARACIÓN
int temperatura;
String tiempo;

//INICIALIZACIÓN
temperatura = 25;
tiempo = "noche";

// SENTENCIAS ANIDADAS
if (tiempo == "noche") {
    if (temperatura >= 26) {
        System.out.println("Es un día caluroso");
    } else if (temperatura >= 22 && temperatura <= 25) {
        System.out.println("Es un día cálido");
    } else if (temperatura >= 17 && temperatura <= 21) {
        System.out.println("Es un día frío");
    } else {
        System.out.println("La temperatura no coincide");
    }
} else
    System.out.println("Es de día");
```



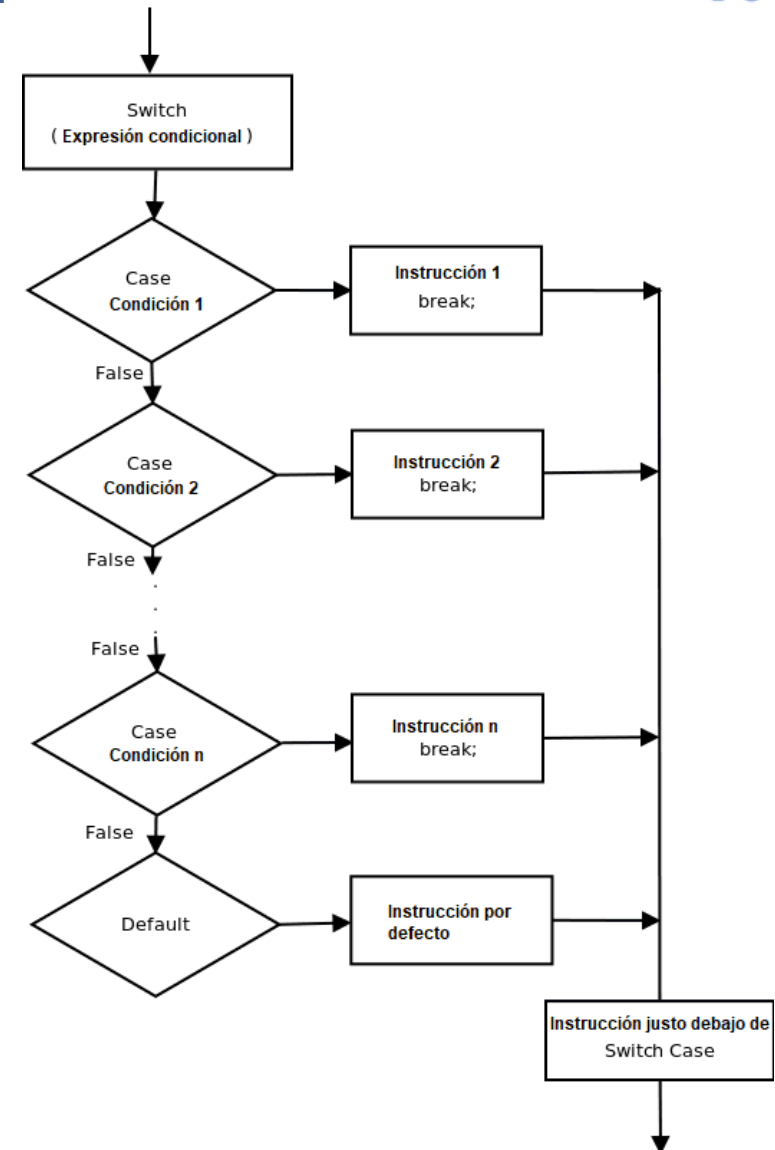
EJERCICIO 01

DECLARAR LA VARIABLE “DIVISION1” CUYO VALOR SEA IGUAL A LA DIVISIÓN ENTRE LA VARIABLE “C” Y “B” RESPECTIVAMENTE; Y VALIDAR SI EL RESULTADO ES UN NUMERO POSITIVO, NEGATIVO, IGUAL A CERO O DIVISIÓN NO POSIBLE.

7. ESTRUCTURAS DE CONDICIÓN

SENTENCIA SWITCH

La sentencia **SWITCH** es una forma de expresión de un anidamiento múltiple de instrucciones if ... else. pero con la diferencia del uso de la anotación llamada **SWITCH Y CASE**, El cual representará un posible resultado en caso cumpla la **EXPRESIÓN CONDICIONAL**. Finalmente, si se cumple una condición debe terminar con la anotación break, la cual indica al programa terminar la evaluación y ejecución.



7. ESTRUCTURAS DE CONDICIÓN

SENTENCIA SWITCH

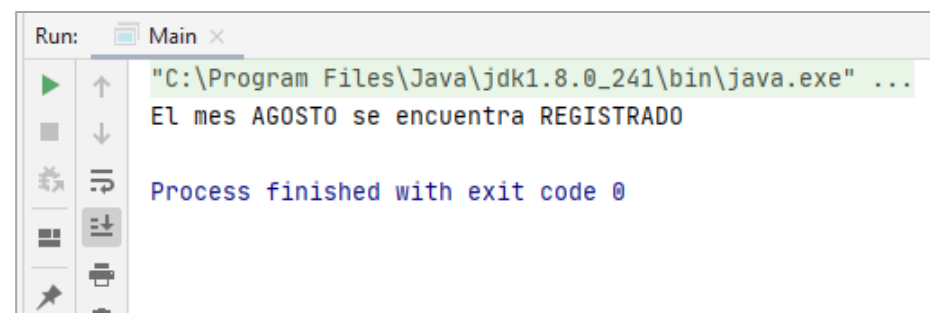
En el siguiente ejemplo demos leerlo de la siguiente manera:

1. En caso el mes ingresado **sea 1**, entonces muestra el mensaje que ENERO se encuentra REGISTRADO.
2. En caso el mes ingresado **sea 3**, entonces muestra el mensaje que MARZO se encuentra REGISTRADO.
3. En caso el mes ingresado **sea 8**, entonces muestra el mensaje que AGOSTO se encuentra REGISTRADO.
4. En caso el mes ingresado **sea 10**, entonces muestra el mensaje que OCTUBRE se encuentra REGISTRADO.
5. En caso el mes ingresado **sea 12**, entonces muestra el mensaje que DICIEMBRE se encuentra REGISTRADO.
6. Default, **sino cumple los casos anteriores**, entonces muestra el mensaje de el MES no se encuentra REGISTRADO.

```
//DECLARACIÓN
int mes;

//INICIALIZACIÓN
mes = 8;

//SENTENCIA SWITCH
switch (mes) { // Condición de selección
    case 1:
        System.out.println("El mes ENERO se encuentra REGISTRADO"); break;
    case 3:
        System.out.println("El mes MARZO se encuentra REGISTRADO");break;
    case 8:
        System.out.println("El mes AGOSTO se encuentra REGISTRADO");break;
    case 10:
        System.out.println("El mes OCTUBRE se encuentra REGISTRADO");break;
    case 12:
        System.out.println("El mes DICIEMBRE se encuentra REGISTRADO");break;
    default:
        System.out.println("El mes no se encuentra registrado");
}
```



```
Run: Main x
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...
El mes AGOSTO se encuentra REGISTRADO

Process finished with exit code 0
```



EJERCICIO 02

Realizar la simulación de una maquina que contiene 5 opciones enumeradas del 0 al 4, en donde la opción 0 es llamar, la opción 1 es enviar mensaje, la opción 2 es apagar, la opción 3 es reiniciar y la opción 4 es autodestruir.

Casos de Prueba:

Opción = 0. Resultado: "Haz marcado la opción Llamar"

Opción = 1. Resultado: "Haz marcado la opción Enviar Mensaje"

Opción = 2. Resultado: "Haz marcado la opción Apagar"

Opción = 3. Resultado: "Haz marcado la opción Reiniciar"

Opción = 4. Resultado: "Haz marcado la opción Autodestruir"

8. ESTRUCTURAS DE ITERACIÓN

SENTENCIA WHILE

La instrucción **WHILE** es un ciclo repetitivo (bucle) basado en los resultados de una expresión lógica. El propósito es repetir un bloque de código mientras una **condición** se mantenga verdadera. Así mismo la **condición** debe dar como resultado un valor booleano, verdadero (**true**) si la condición se cumple, o falso si esta no se cumple (**false**).

El siguiente ejemplo podemos leerlo de la siguiente manera:

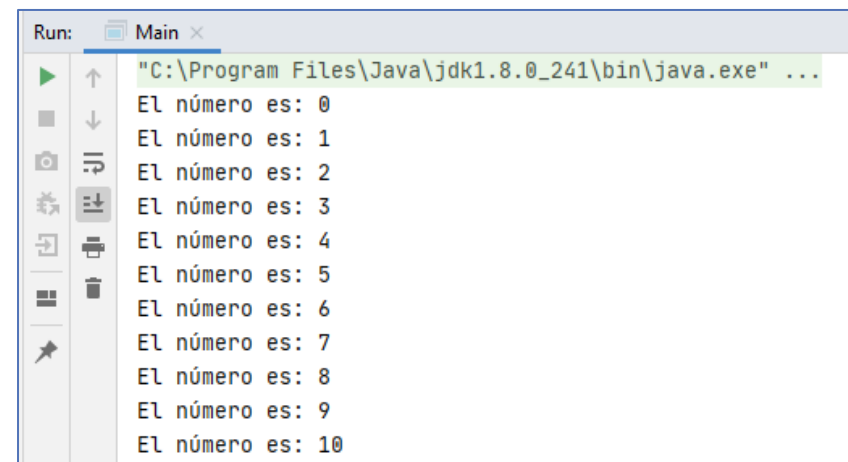
Mientras la variable **inicio** sea **menor o igual** a 10. Entonces **imprimes** el valor actual del número y finalmente incrementar la variable **inicio** en 1 **hasta** que se cumpla la condición del **WHILE**.

```
//DECLARACIÓN
int inicio;

//INICIALIZACIÓN
inicio = 0;

// INSTRUCCIÓN WHILE
while (inicio <= 10) {
    System.out.println("El número es: " + inicio);
    inicio++;
}
```

Verifica la condición (<= 10) ANTES de actualizar el valor (Inicio++)



```
Run: Main x
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...
El número es: 0
El número es: 1
El número es: 2
El número es: 3
El número es: 4
El número es: 5
El número es: 6
El número es: 7
El número es: 8
El número es: 9
El número es: 10
```


EJERCICIO 03

Declarar una variable flotante (v1) y dividirla entre otra variable (v2) flotante hasta que su valor sea menor o igual a 1. Utilizar la sentencia While e imprimir el valor de la primera variable (v1) en cada iteración.

Casos de Prueba:

v1 = 50, v2 = 3. Salida: "16.666666", "5.5555553", "1.8518518", "0.61728394"

v1 = 100.6, v2 = 4.5. Salida: "22.355555", "4.967901", "1.103978", "0.24532846"

8. ESTRUCTURAS DE ITERACIÓN

DO ... WHILE

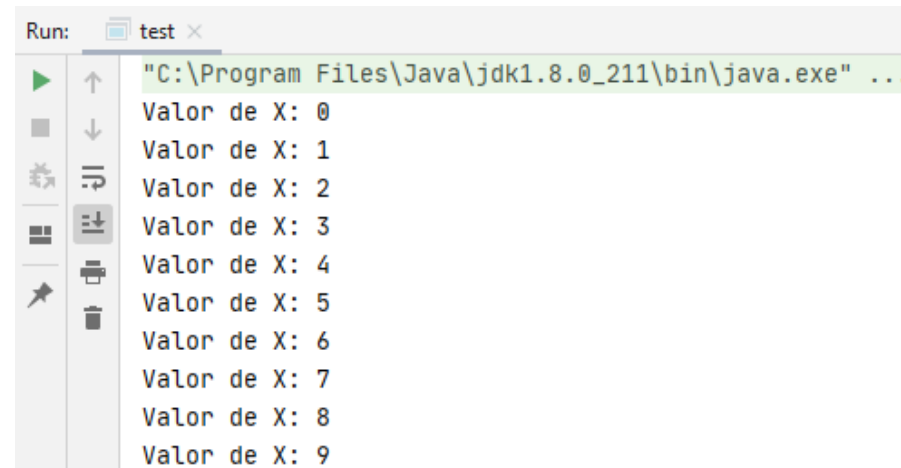
La instrucción DO WHILE es similar al WHILE con la única diferencia de que comprueba la condición DESPUES de ejecutar las instrucciones.

1. El bucle do while comienza con la ejecución de la(s) declaración(es). No hay verificación de ninguna condición la primera vez.
2. Después de la ejecución, y la actualización del valor de la variable, la condición se verifica para el verdadero o falso. Si se evalúa como verdadero, comienza la siguiente iteración del ciclo.
3. Cuando la condición se vuelve falsa, el ciclo finaliza y marca el final de su ciclo de su vida.
4. Es importante tener en cuenta que el bucle Do While ejecutará sus declaraciones al menos una vez antes de que se verifique cualquier condición, y por lo tanto es un ejemplo de bucle de control de salida.

Verifica la condición
($x < 10$) DESPUES de
Actualizar el valor
($x++$)

```
//DECLARACIÓN y ASIGNACIÓN DE VALORES
int x = 0;

do{
    // EL CÓDIGO DEL "DO" SE IMPRIME INCLUSIVE
    // SI LA CONDICIÓN ES FALSA
    System.out.println("Valor de X: "+x);
    x++; //INCREMENTA x de 1
}while(x < 10);
```



Run: test x

"C:\Program Files\Java\jdk1.8.0_211\bin\java.exe" ...

Valor de X: 0
Valor de X: 1
Valor de X: 2
Valor de X: 3
Valor de X: 4
Valor de X: 5
Valor de X: 6
Valor de X: 7
Valor de X: 8
Valor de X: 9

EJERCICIO 04

Crear un programa que pida al usuario su contraseña (numérica). Deberá terminar cuando introduzca como contraseña el número 4567, pero volvérsela a pedir tantas veces como sea necesario. Cada vez que ingrese una contraseña falsa imprimir “Inválido” y cuando ingrese la contraseña correcta imprimir “Válido”

Casos de Prueba:

clave = 710, 125, 685, 4567. Resultado: “Inválido, Inválido, Inválido, Válido”

clave = 4567. Resultado: “Válido ”

Trucos :

Para imprimir un mensaje al usuario utilizar : `System.out.println();`

Para recuperar el valor ingresado, usar : `scanner.nextLine();` , y guardar el resultado en una variable. Hay un ejemplo de uso en este curso...

8. ESTRUCTURAS DE ITERACIÓN

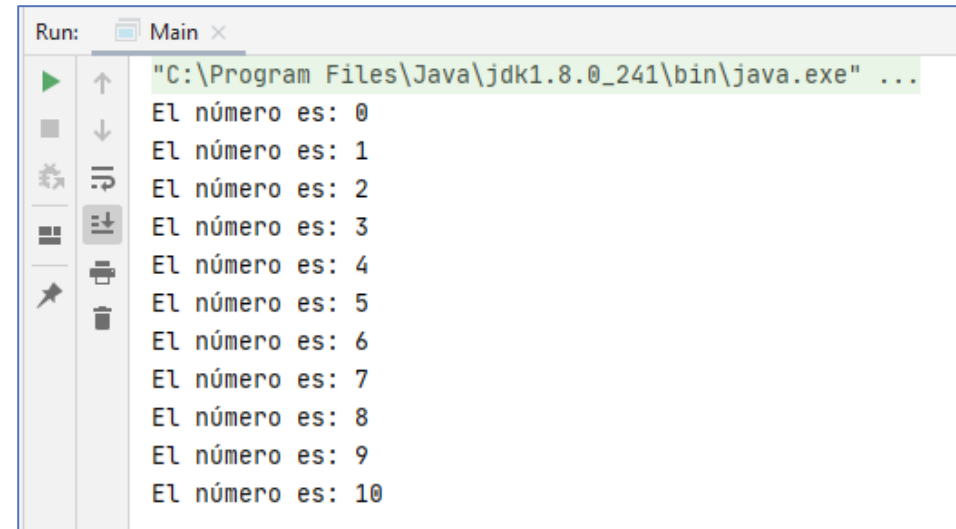
FOR

La instrucción FOR es un ciclo repetitivo de igual manera que el WHILE, a diferencia que ahora tendremos el control de la inicialización, condición y conocimiento de la instrucción que detendrá el ciclo repetitivo, todo esto dentro del mismo método FOR.

El siguiente ejemplo podemos leerlo de la siguiente manera:

1. Dado el ciclo repetitivo FOR, que empezará con la variable inicio con el valor 0, entonces se ejecutará el ciclo mientras la variable inicio sea menor o igual a 10, y por cada iteración la variable inicio incrementará en 1

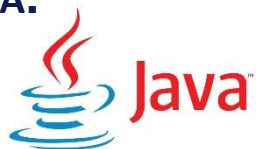
```
//DECLARACIÓN  
int inicio;  
  
for (inicio = 0; inicio <= 10; inicio++)  
    System.out.println("El número es: " + inicio);  
}
```



Run: Main x

```
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...  
El número es: 0  
El número es: 1  
El número es: 2  
El número es: 3  
El número es: 4  
El número es: 5  
El número es: 6  
El número es: 7  
El número es: 8  
El número es: 9  
El número es: 10
```

SE OBTIENE EL MISMO RESULTADO QUE EL EJEMPLO WHILE, PERO
CON UN CONTROL DISTINTO EN SU ESTRUCTURA.



EJERCICIO 05

En matemáticas, la sucesión o serie de Fibonacci es la siguiente sucesión infinita de números naturales.
0,1,1,2,3,5,8,13,21,34,55,89,144,...

La sucesión comienza con los números 0 y 1 y a partir de estos cada término es la suma de los dos anteriores, es la relación de recurrencia que la define.

EJERCICIO PRÁCTICO 6: (6 PUNTOS) – 15 MINUTOS

Imprimir los N primeros números de la secuencia Fibonacci.

Casos de Prueba:

N = 5. Salida: 0,1,1,2,3

N = 10. Salida: 0,1,1,2,3,5,8,13,21,34

9. CADENAS DE CARACTERES

CHAR: Caracter

Una variable de tipo **char** almacena 1 carácter Unicode y solamente uno. El tipo char se expresa encerrando el carácter entre **comillas simples**.



```
//DECLARACIÓN
char caracter;
//ASIGNACIÓN DE VALORES
caracter = 'z';
//IMPRIMIR EL VALOR
System.out.println(caracter);
```

```
Run: test x
"C:\Program Files\Java\jdk1.8.0_211\bin\java.exe" .
z
Process finished with exit code 0
```

STRING: Cadena de caracteres

Una cadena es una **secuencia de caracteres**. Para el manejo de cadenas, java provee una clase llamada **String**. Esta clase, es la representación como objeto de un arreglo de caracteres que no se puede cambiar. Se expresa encerrando la secuencia entre **comillas dobles**.



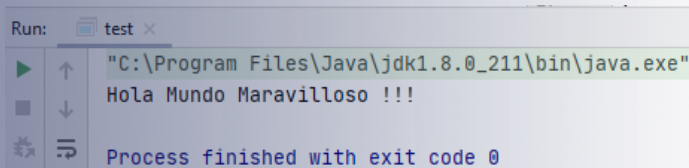
```
//DECLARACIÓN
String cadena;
//ASIGNACIÓN DE VALORES
cadena = "Es una cadena de caracteres.";
//IMPRIMIR EL VALOR
System.out.println(cadena);
```

```
Run: test x
"C:\Program Files\Java\jdk1.8.0_211\bin\java.exe" .
Es una cadena de caracteres.
Process finished with exit code 0
```


9. CADENAS DE CARACTERES

CONCATENACIÓN

```
//DECLARACIÓN
String cadena1;
String cadena2;
String resultado;
//ASIGNACIÓN DE VALORES
cadena1 = "Hola Mundo";
cadena2 = " Maravilloso !!!";
//CONCATENACIÓN DE 2 CADENAS
resultado = cadena1 + cadena2;
//IMPRIMIR LA CADENA RESULTADO
System.out.println(resultado);
```



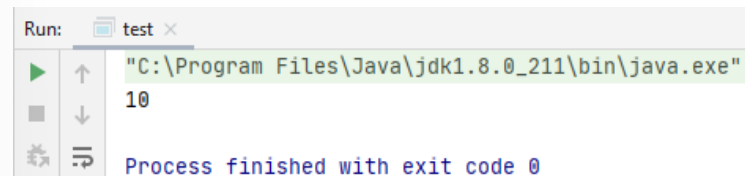
Run: test x

```
"C:\Program Files\Java\jdk1.8.0_211\bin\java.exe"
Hola Mundo Maravilloso !!!
Process finished with exit code 0
```

EL OPERADOR ES: “ + ”

LONGITUD

```
//DECLARACIÓN
String cadena;
//ASIGNACIÓN DE VALORES
cadena = "Hola Mundo";
//IMPRIMIR LA CADENA RESULTADO
System.out.println(cadena.length());
```



Run: test x

```
"C:\Program Files\Java\jdk1.8.0_211\bin\java.exe"
10
Process finished with exit code 0
```

EL OPERADOR ES: “ length() ”

COMPARACIÓN

```
//DECLARACIÓN y ASIGNACIÓN DE VALORES
String cadena1 = "Hola";
String cadena2 = "Mundo";
//Si las cadenas son idénticas
if (cadena1.equals(cadena2))
{
    //Ambas cadenas son iguales (no es el caso)
    System.out.println("Son idénticas");
}
else
{
    //Las cadenas son diferentes (es el caso)
    System.out.println("Son diferentes");
}
```

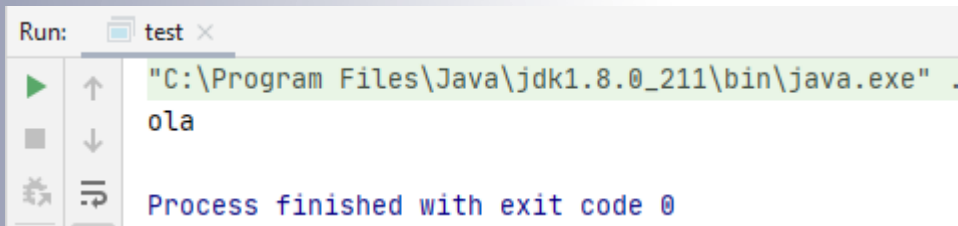
EL OPERADOR ES: “ () ”



9. CADENAS DE CARACTERES

EXTRAER UN APARTE DE UNA CADENA

```
//DECLARACIÓN y ASIGNACIÓN DE VALORES
String cadena = "Hola Mundo";
String resultado = "";
//ASIGNAR UNA PARTE DE LA CADENA A LA CADENA RESULTADO
//DESDE EL 2ndo CARACTER al 4to CARACTER
resultado = cadena.substring(1,4);
//IMPRIMIR RESULTADO
System.out.println(resultado);
```



```
Run: test x
"C:\Program Files\Java\jdk1.8.0_211\bin\java.exe" .
ola
Process finished with exit code 0
```

UNA COSA MUY IMPORTANTE ES QUE LA PRIMERA LETRA DE UNA CADENA DE TEXTO SIEMPRE TIENE EL ÍNDICE 0.

Estas serían las posiciones de la frase «Hola Mundo».

H	o	l	a		M	u	n	d	o
0	1	2	3	4	5	6	7	8	9

SE USA EL MÉTODO :
Substring (int indexInicio, int indexFin)



EJERCICIO 06

Declarar una cadena y asignarle una frase con al menos 3 palabras.
Asignar cada palabra de la frase a una nueva cadena (se tendrá que declarar una cadena por cada palabra.
Imprimir cada palabra por separado, y su longitud.

Ejemplo: Hola Mundo Feliz

Hola

4

Mundo

5

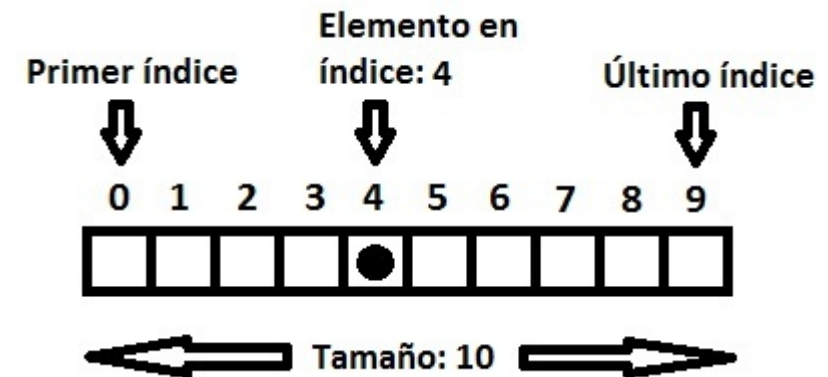
Feliz

5

10. ARREGLOS

LOS **ARREGLOS** SE PUEDEN DEFINIR COMO OBJETOS EN LOS QUE PODEMOS GUARDAR MAS DE UNA VARIABLE, ES DECIR, AL TENER UN ÚNICO ARREGLO, ESTE PUEDE GUARDAR MÚLTIPLES VARIABLES DE ACUERDO A SU TAMAÑO O CAPACIDAD, **ES IMPORTANTE RECORDAR QUE LAS VARIABLES GUARDADAS DEBEN SER DEL MISMO TIPO**, POR EJEMPLO: SI TENEMOS UN ARREGLO DE TIPO NUMÉRICO QUE PUEDE ALMACENAR 10 VARIABLES, SOLO PODRÁ ALMACENAR 10 NÚMEROS DIFERENTES, NO OTRAS VARIABLES COMO CARACTERES O STRINGS.

UN ARREGLO SE PUEDE CREAR CON VALORES PREDETERMINADOS O INGRESARLOS DE MANERA DINÁMICA DURANTE LA EJECUCIÓN.

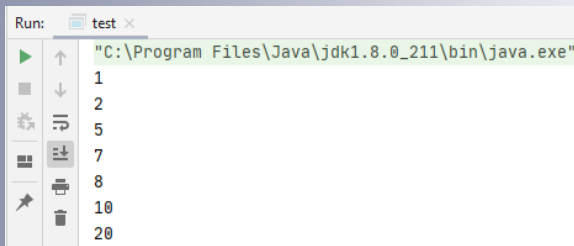


10. ARREGLOS

CON VALORES PREDETERMINADOS

```
//DECLARACIÓN
//Arreglo con valores predefinidos
int arreglo[] = {1,2,5,7,8,10,20};

//Se imprime cada valor del arreglo
// del 1er índice al último
for (int i=0; i<arreglo.length; i++)
{
    System.out.println(arreglo[i]);
}
```



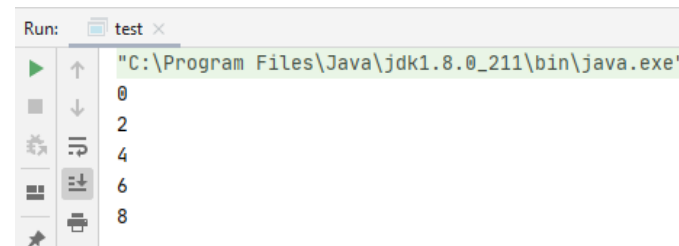
```
Run: test x
"C:\Program Files\Java\jdk1.8.0_211\bin\java.exe"
1
2
5
7
8
10
20
```

AL IMPRIMIR LOS VALORES DEL ARREGLO, SE OBSERVA QUE IMPRIME TODOS LOS VALORES EN EL ORDEN QUE SE DECLARO EN EL PROGRAMA.

```
//DECLARACIÓN con Tamaño de 5
int arreglo[] = new int [5];

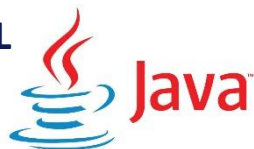
//ASIGNACIÓN DE VALORES
// (i * 2 es decir 0, 2, 4, 6 , 8)
for (int i=0; i<arreglo.length; i++)
{
    arreglo[i] = i * 2;
}

//SE IMPRIME CADA VALOR
for (int i=0; i<arreglo.length; i++)
{
    System.out.println(arreglo[i]);
}
```



```
Run: test x
"C:\Program Files\Java\jdk1.8.0_211\bin\java.exe"
0
2
4
6
8
```

AL IMPRIMIR LOS VALORES DEL ARREGLO, SE OBSERVA QUE IMPRIME TODOS LOS VALORES GENERADOS POR LA MULTIPLICACIÓN REALIZADA DURANTE LA EJECUCIÓN DEL PROGRAMA.



EJERCICIO 07

Dado un arreglo de notas que están entre 0 – 20.
Calcular el promedio simple de todas las notas e imprimirlo en pantalla.

Casos de Prueba:

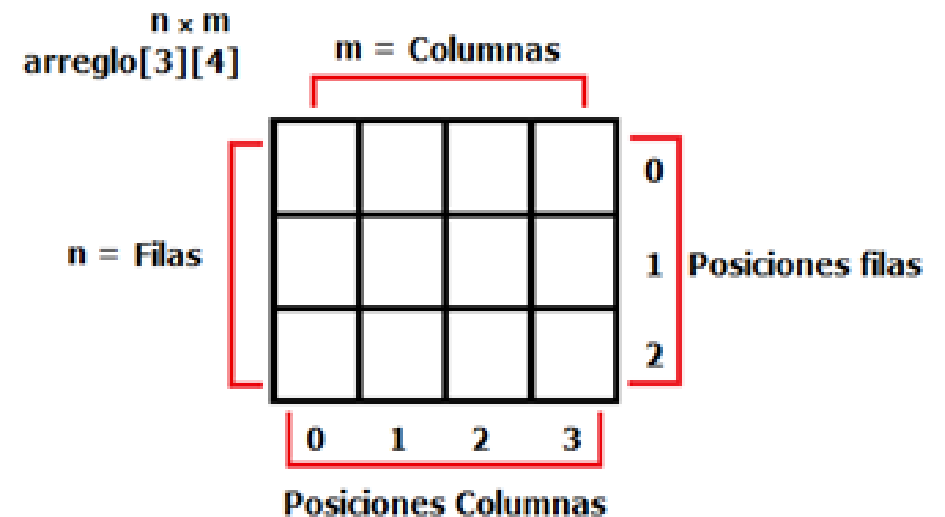
1. Arreglo = {10,12,15,20}. Resultado: “El promedio es 14.25”
2. Arreglo = {15,10}. Resultado: “El promedio es 12.5”
3. Arreglo = {20,0,15,13,17}. Resultado: “El promedio es 13”
4. Arreglo = {0,0,15,13,10}. Resultado: “El promedio es 7.6”

11. MATRICES

LOS **ARREGLOS BIDIMENSIONALES** TAMBIÉN LLAMADOS MATRICES O **TABLAS**, SON UN TIPO DE ARREGLO QUE ALMACENA UNA LISTA DE ARREGLOS DENTRO DE ELLA.

LA DIMENSIÓN DE UN ARREGLO LA DETERMINA EL NÚMERO DE ÍNDICES NECESARIOS PARA ACCEDER A SUS ELEMENTOS.

UNA MATRIZ NECESITA **DOS** ÍNDICES PARA ACCEDER A SUS ELEMENTOS. GRÁFICAMENTE PODEMOS REPRESENTAR UNA MATRIZ COMO UNA TABLA DE **N** FILAS Y **M** COLUMNAS CUYOS ELEMENTOS SON TODOS DEL MISMO TIPO.

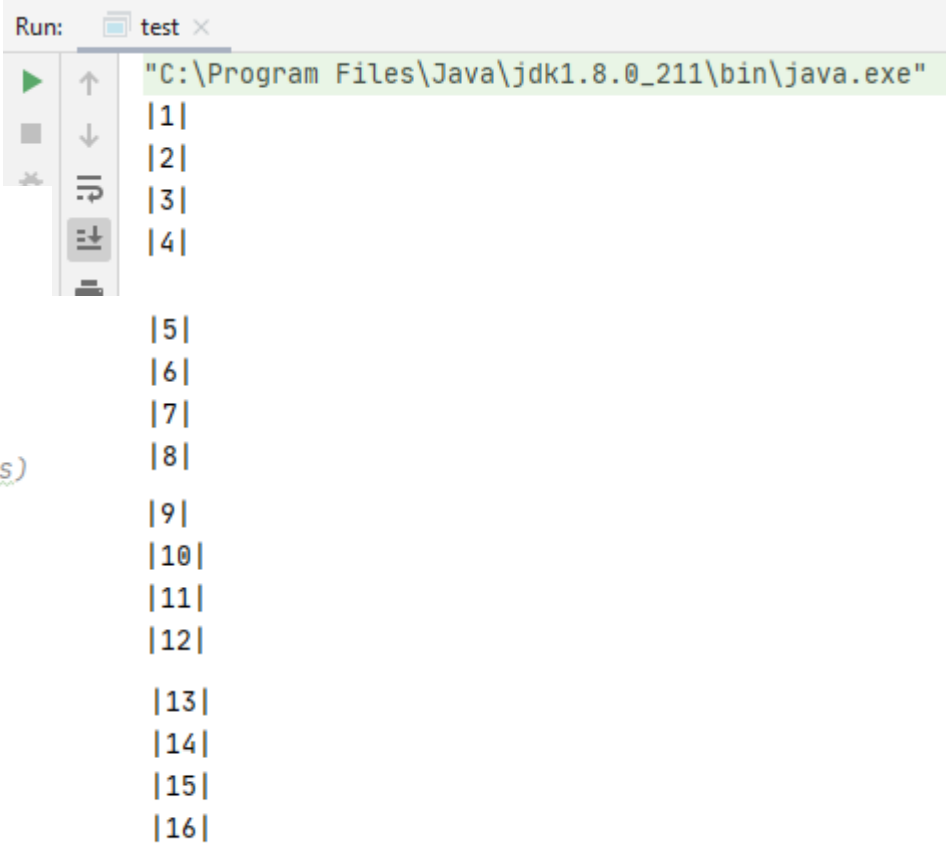


11. MATRICES

CON VALORES PREDETERMINADOS

```
//DECLARACIÓN y ASIGNACIÓN DE VALORES
int matrice[][] = { {1,2,3,4} , {5,6,7,8} , {9,10,11,12} , {13,14,15,16} };

//SE IMPRIME CADA VALOR
for (int i=0; i<matrice.length; i++) //recorre los arreglos principales (lineas)
{
    for (int j=0; j<matrice[i].length; j++) //recorre los arreglos secundarios (columnas)
    {
        System.out.println("|" + matrice[i][j] + "|");//imprimir 1 valor
    }
    System.out.println(""); //salto de linea
}
```



```
Run: test x
"C:\Program Files\Java\jdk1.8.0_211\bin\java.exe"
|1|
|2|
|3|
|4|
|5|
|6|
|7|
|8|
|9|
|10|
|11|
|12|
|13|
|14|
|15|
|16|
```

AL IMPRIMIR SE OBSERVA QUE IMPRIME LOS VALORES DE CADA COLUMNA, PARA CADA LÍNEA.



EJERCICIO 08

Dado un arreglo de notas que están entre 0 – 20.

Dada la siguiente matriz:

	0	1	2	3	4
0	1	2	3	4	5
1	2	4	6	8	10
2	3	6	9	12	15

Imprimir el mayor número y la fila y columna (posición) donde se ubica.

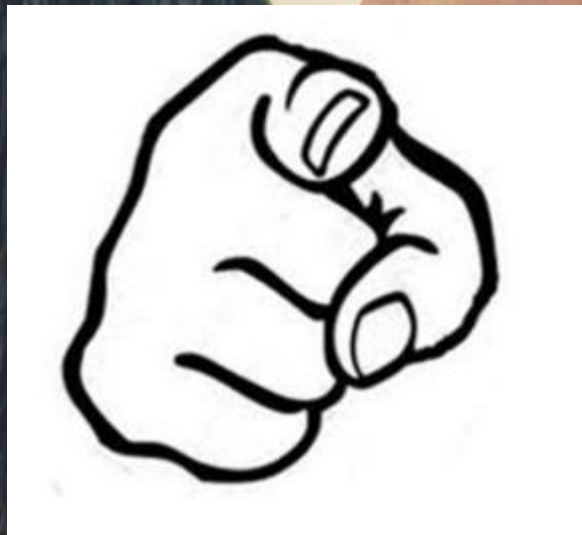
Imprimir el menor número y la fila y columna (posición) donde se ubica.

Resultado esperado:

“El mayor número es 15 y se encuentra en la fila 2, columna 4”

“El menor número es 1 y se encuentra en la fila 0, columna 0”

TE TOCA A TI



TE TOCA A TI

A woman with reddish-brown hair is shown in profile, interacting with a large, vertical digital screen. The screen displays various data visualizations, including a circular gauge and a tree-like structure. Above the screen, two glowing, wireframe butterflies are visible, surrounded by a cloud of small, glowing particles. The background is a dark, out-of-focus city street at night, with warm lights from buildings and streetlights. The overall scene has a futuristic, high-tech feel.

NTT DATA

GRACIAS

**FUTURE
AT HEART**