

# Java Inicial

(20 horas)





# Temario

1. Programación Orientada a Objetos
2. Introducción y Sintaxis Java
3. **Sentencias Control Flujo**
4. POO en Java
5. Relaciones entre Objetos
6. Polimorfismo, abstracción e interfaces
7. Excepciones
8. Conceptos avanzados



# Tema 3

## Sentencias de Control de Flujo



# Objetivos

1. Programación Orientada a Objetos
  2. Introducción y Sintaxis Java
  3. **Sentencias Control Flujo**
  4. POO en Java
  5. Relaciones entre Objetos
  6. Polimorfismo, abstracción e interfaces
  7. Excepciones
  8. Conceptos avanzados
- Sentencias condicionales
    - IF... ELSE
    - SWITCH
    - OPERADOR TERNARIO ?
  - Sentencias Iterativas (Bucles)
    - WHILE
    - DO ... WHILE
    - FOR
  - Sentencias de salto incondicional
    - Break
    - Continue
    - Return

- Sentencias de control de flujo
  - Dentro de nuestro programa las sentencias se ejecutan de manera secuencial.
  - A veces necesitamos que se ejecutó sólo una parte del código o tras alguna condición.
  - Para ello java proporciona **Sentencias control flujo.**
    - Controlar de una manera más eficaz la ejecución de nuestros programas.

- Sentencias de control de flujo
  - Se basan en la evaluación de una expresión.
  - En función del resultado, ejecutar un bloque de código u otro.
  - Tres tipos
    - If ... else ...
    - Switch
    - Operador Ternario ?

## ■ IF ... ELSE...

- Si la **expresión Condicional** se evalúa a **true** se ejecutará el bloque asociado con la parte **if**. Por el contrario, si se evalúa a **false**, se ejecutará el bloque asociado con la parte **else**.
- La parte del else es opcional

```
SENTENCIA IF ... ELSE  
Su sintaxis más simple es la siguiente:  
  
if ( expresionCondicional )  
{  
    sentencia 1;  
    ...  
    sentencia N;  
}  
[ else  
  {  
    sentencia 1;  
    ...  
    sentencia N;  
  } ]
```

- Ejercicio: ¿son expresiones condicionales?
  - boolean b= 1 < 5
  - boolean b=(8+3 ) \* 11
  - boolean b= true || false
  - boolean b=(true || false) || (true & false)
  - boolean b= true + false

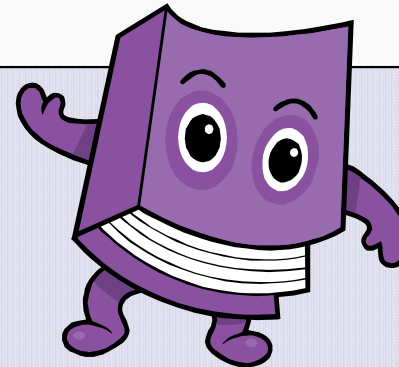
## ■ Ejercicio práctico:

### □ Ver si un año es bisiesto

```
if ((( año % 4 == 0 ) && ( año % 100 != 0 )) || ( año % 400 == 0 )) {  
    System.out.println("Es bisiesto");  
} else {  
    System.out.println("No es bisiesto");  
}
```

```
    }  
    } else {  
        System.out.println("Es bisiesto");  
    }  
} else {  
    System.out.println("No es bisiesto");  
}
```

■ .... Suerte!!!



## ■ IF ... ELSE... (*Conjunción/Disyunción*)

### Conjunción

```
if (condición1 && condición2){  
    sentencial;  
} else {  
    sentencia2;  
}
```

```
if ( condición1 ) {  
    if ( condición2 ) {  
        sentencial;  
    } else {  
        sentencia2;  
    }  
} else {  
    sentencia2;  
}
```

### Disyunción

```
if ( condición1 || condición2 ) {  
    sentencial;  
} else {  
    sentencia2;  
}
```

```
if ( condición1 ){  
    sentencial;  
} else {  
    if ( condición2 ) {  
        sentencial;  
    } else {  
        sentencia2;  
    }  
}
```

## ■ IF ... ELSE... (*Negación*)

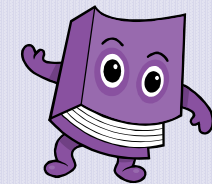
Negación

```
if ( ! condición1) {  
    sentencia1;  
} else {  
    sentencia2;  
}
```

```
if ( condición1) {  
    sentencia2;  
} else {  
    sentencia1;  
}
```

## ■ Ejercicio práctico:

- Ejercicio: detectar si dado un número, éste es par o impar y mostrarlo por pantalla.
- Ejercicio: dados dos números, detectar cual de los dos es mayor y escribirlo por pantalla.
- Ejercicio: dados 5 números, detectar cual de los 5 es mayor y escribirlo por pantalla.
- Ejercicio: calculadora: programar una aplicación que, dada una operación (+, -, \*, /) realice esa operación con dos números datos.



■ .... Suerte!!!

## ■ Switch

- Sustituye a muchos if ... else anidados
- La **expresión** debe evaluarse a un valor numérico entero o char.
- Sus valores los comparamos con cada *case*.
- En caso de igualdad se ejecuta **desde ahí**
- La parte *default* es optativa
  - Se ejecuta si la expresión no es igual al resto de constantes anteriores.
- Se puede usar un *break*
  - Para evitar continuar ejecutando el resto de código

```
switch (expresión)
{
    case CONSTANTE_1:
        sentencial
        ...
        [ break; ]
    ...
    case CONSTANTE_N:
        sentencial
        ...
        [ break; ]
    [default:
        sentencial
        ...]
}
```

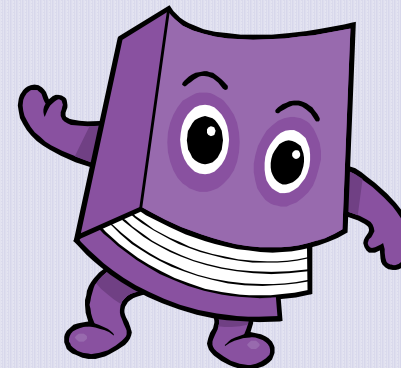
## ■ Switch: Ejemplo

```
class TestSwitch2
{
    public static void main(String args[])
    {
        int mes=Integer.parseInt(args[0]);
        System.out.print ("El mes " + mes + " es del: ");
        switch(mes)
        {
            case 1:
            case 2:
            case 3:
                System.out.println ("PRIMER TRIMESTRE");
                break;
            case 4:
            case 5:
            case 6:
                System.out.println ("SEGUNDO TRIMESTRE");
                break;
            case 7:
            case 8:
            case 9:
                System.out.println ("TERCER TRIMESTRE");
                break;
            case 10:
            case 11:
            case 12:
                System.out.println ("CUARTO TRIMESTRE");
                break;
            default:
                System.out.println ("ERROR");
        }
    }
}
```

## ■ Ejemplo workspace:

### □ EjemploCase.java

- Este ejemplo muestra cómo utilizar el switch...case con chars.

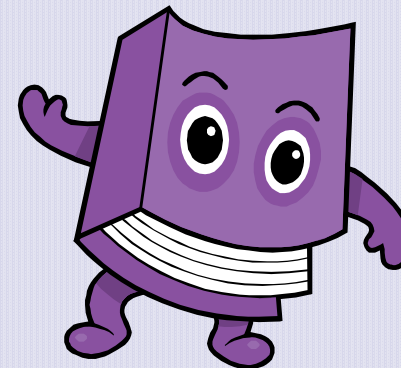


## ■ Ejercicio práctico:

### □ Minicalculadora versión 2

```
public class MiniCalculadora{
    public static void main(String args[]){
        int a = 1;
        int b = 1;
        char op = '/';
        System.out.print("El resultado es : ");
        switch ( op ) {
            case '+':
                System.out.println( a + b );
                break;
            case '-':
                System.out.println( a - b );
                break;
            case '*':
                System.out.println( a * b );
                break;
            case '/':
                System.out.println( a / b );
                break;
            default:
                System.out.println("error" );
                break;
        }
    }
}
```

■ .... Suerte!!!



## ■ Operador ternario “? : ”

- La ***expresion1*** se evaluará siempre que ***expresionCondicional*** sea ***true***.
- En otro caso se evaluará ***expresion2***.

```
expresionCondicional ? expresion1 : expresion2
```

## □ Ejemplo

```
...  
if (x < 10)  
    y = x - 1;  
else  
    y = x + 1;  
...
```

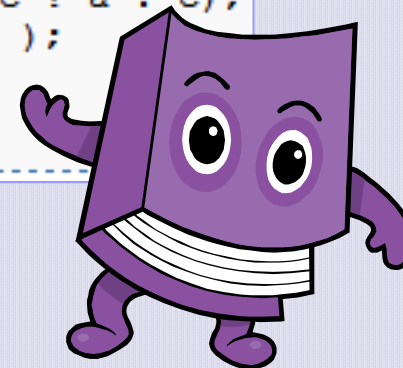
## ■ Ejercicio práctico:

- Mirar cual es el medio de tres numeros (a,b,c)

```
Si A > B, entonces  
X tal que X > A y X < B
```

```
public class NumeroMedio{  
    public static void main(String args[]){  
        int a = 2 ;  
        int b = 12;  
        int c = 3;  
        int j = a > b ? ( b > c ? b : c ) : ( a > c ? a : c );  
        System.out.println("El resultado es: " + j );  
    }  
}
```

■ .... Suerte!!!



## ■ Sentencias iterativas o Bucles

- Necesarias para repetir partes del código
  - Numero fijo de veces.
  - Dependiendo de la evaluación de una condición
- Tres tipos
  - *WHILE*
  - *DO... WHILE*
  - Bucle *FOR*

## ■ Sentencia While

- Evaluamos ***expresionCondicional*** y si su resultado es ***true*** procederá a ejecutar el bloque asociado a esta sentencia, procediendo a una nueva evaluación de la ***expresión Condicional*** para ver si de nuevo ejecuta el bloque o no.

**SENTENCIA WHILE**  
Su sintaxis es la siguiente:

```
while ( expresionCondicional )  
{  
    sentencia;  
    ...  
    sentenciaN;  
}
```

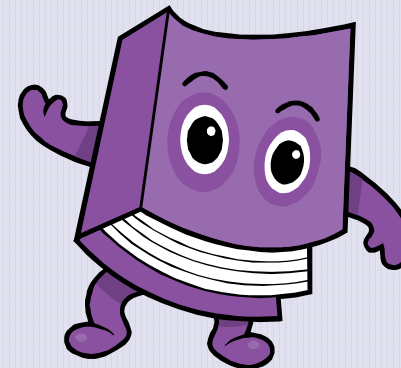
- Ejemplo: Factorial de un número n

```
while(n > 0)  
{  
    resultado = resultado * n;  
    n--;  
}  
System.out.println (resultado);
```

## ■ Ejemplo workspace:

### □ EjemploWhile.java

- En este ejemplo se muestra cómo realizar un bucle sencillo con la sentencia while hasta que deja de cumplirse una condición.



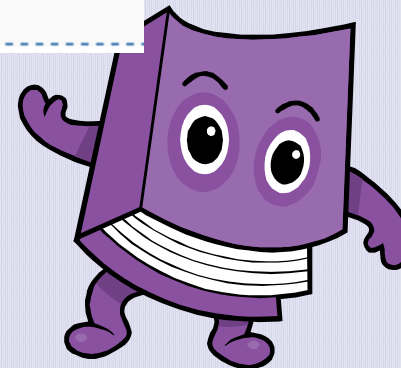
## ■ Ejercicio práctico:

### □ Ejemplo While

```
boolean prueba = true;
while ( prueba ) {
    System.out.println("Esto lo verás una vez");
    prueba = false;
}
```

```
boolean prueba = true;
while ( prueba ) {
    System.out.println("Esto lo verás muchas veces");
}
```

■ .... Suerte!!!

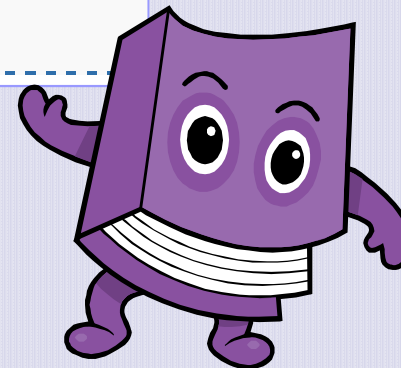


## ■ Ejercicio práctico:

- Pintar un Cuadrado de 5 "\*\*\*\*\*"

```
public class Cuadrado{
    public static void main(String args[]){
        int contador = 1;
        while ( contador <= 5 ) {
            System.out.println("*****");
            contador++;
        }
    }
}
```

- .... Suerte!!!



## ■ DO ... WHILE

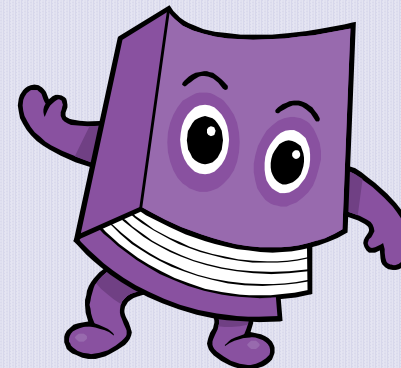
- Parecido al While pero siempre se ejecuta
- La *expresión* se ejecuta posterior a la ejecución del bloque asociado.

```
class TestDoWhile
{
    public static void main(String args[])
    {
        int numeros[]={3,4,2,11,15};
        int i = 0;
        do
        {
            System.out.print (2*numeros[i] + " ");
            i++;
        } while(i < numeros.length);
    }
}
```

## ■ Ejemplo workspace:

### □ EjemploDoWhile.java

- En este ejemplo se muestra cómo realizar un bucle con la sentencia do...while hasta que deja de cumplirse una condición. En este caso recuerda que se ejecuta al menos una vez.

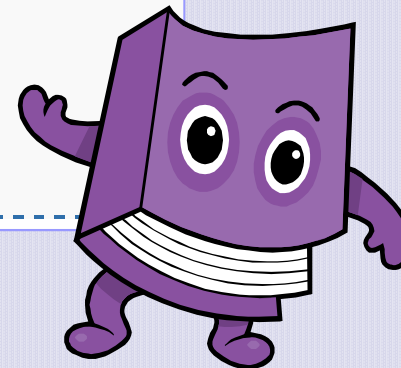


## ■ Ejercicio práctico:

### □ Contador de dígitos

```
public class CuentaDigitos{
    public static void main(String args[]){
        int número = 4557888;
        int dígitos = 0;
        do {
            número /=10;
            dígitos++;
        }
        while ( número > 0 );
        System.out.println(dígitos);
    }
}
```

■ .... Suerte!!!



## ■ FOR

```
class TestFor
{
    public static void main(String args[])
    {
        int numeros[] = {3,5,7,2,12};
        int i;
        for(i=0;i<numeros.length;i++)
            System.out.print (numeros[i]*2 + " ");
    }
}
```

- *Inicialización* es la zona donde se inicializa el contador de iteraciones.
- *Condición* indica si se debe ejecutar el bloque asociado.
- *Incremento* es la zona que se ejecuta tras el bloque asociado, normalmente incrementando o decrementando el contador de iteraciones.

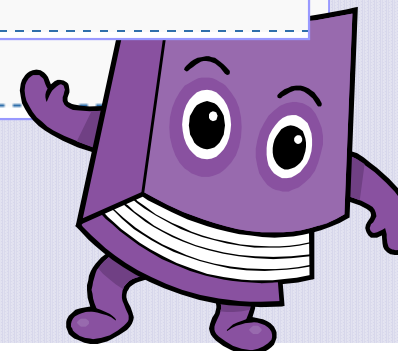
## ■ Ejercicio práctico:

- La tabla de multiplicar del 3

```
System.out.println("3 x 1 = 3");  
System.out.println("3 x 2 = 6");  
System.out.println("3 x 3 = 9");  
System.out.println("3 x 4 = 12");  
System.out.println("3 x 5 = 15");  
System.out.println("3 x 6 = 18");  
System.out.println("3 x 7 = 21");  
System.out.println("3 x 8 = 24");  
System.out.println("3 x 9 = 27");
```

```
for ( int factor = 1; factor <= 9; factor ++ ) {  
    System.out.println("3 x " + factor + " = " + 3*factor );  
}  
}
```

■ .... Suerte!!!



## ■ Sentencias de salto incondicional

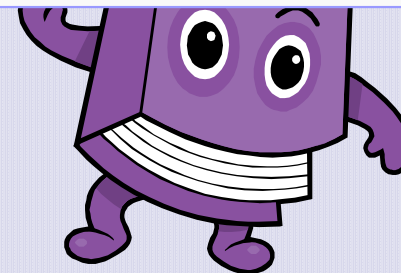
- No es recomendable.
- Su objetivo es la finalización de ejecución de un bloque.
- **break**: Rompe la ejecución de un bloque.
- **continue**: Obliga a una nueva iteración de una sentencia de tipo iterativa
  - Comprobando la condición (en el caso de *while* o *for*)
  - Ejecutando la primera sentencia del bloque asociado al bucle (en el caso de *do..while*)
- **return**: Rompe la ejecución de una función (método).
  - Suele venir asociado con un valor que coincide con el tipo de retorno de la función.

## ■ Ejercicio práctico:

- Contador de dígitos hasta 5

```
public class CuentaDigitos{
    public static void main(String args[]){
        int número = 4557888;
        int dígitos = 0;
        while ( número > 0 ) {
            número /=10;
            dígitos++;
            if (dígitos ==5) break;
        }
        if (dígitos ==5) System.out.println("El numero tiene 5 o más dígitos");
    }
}
```

■ .... Suerte!!!





# Conclusiones

1. Programación Orientada a Objetos
2. Introducción y Sintaxis Java
3. **Sentencias Control Flujo**
4. POO en Java
5. Relaciones entre Objetos
6. Polimorfismo, abstracción e interfaces
7. Excepciones
8. Conceptos avanzados

- Sentencias condicionales
  - IF... ELSE
  - SWITCH
  - OPERADOR TERNARIO ?
- Sentencias Iterativas (Bucles)
  - WHILE
  - DO ... WHILE
  - FOR
- Sentencias de salto incondicional
  - Break
  - Continue
  - Return